UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

APRENDIZAGEM AUTOMÁTICA EM LARGA ESCALA NAS REDES SOCIAIS PARA A DESCOBERTA DE AMEAÇAS DE SEGURANÇA

Mestrado em Segurança Informática

André Marques Correia

Dissertação orientada por:
Prof. Doutor Pedro Miguel Frazão Fernandes Ferreira e Prof. Doutor Alysson Neves Bessani

2016

Agradecimentos

Começo por agradecer à minha família por terem acreditado em mim e me apoiarem durante toda a vida.

Ao Professor Pedro Ferreira e ao Professor Alysson Bessani por toda a ajuda que deram para elaborar este projeto. Sem a paciência e disponibilidade dos Professores não teria chegado tão longe.

Eu quero agradecer aos meus colegas da FCUL, principalmente ao Branca, à Copy, à DaCosta, ao GPS, ao Maias, à Paste, ao Valeta e a todos os que não mencionei da Legião1112 mas que foram igualmente importantes. Com vocês passei dos melhores momentos da minha vida e que me fizeram a pessoa que sou atualmente. Todos os momentos e aventuras, o que sofremos e celebrámos, o que aprendemos e ensinámos, nunca será esquecido. Agradeço aos mais velhos por tudo que fizeram por mim e aos mais novos pela felicidade que me trouxeram.

Ainda na FCUL, agradeço ao pessoal da sala 25 e ao pessoal da UI com quem trabalhei, pois graças à experiência que passei tornei-me uma pessoa mais responsável e preparada para o futuro.

Um grande abraço aos dois chatos que mais trabalharam comigo e à bióloga. Ao Cobertor, à Floribela e à Inês. Graças a vocês, bebi uma quantidade absurda de cafés, tive discussões filosóficas profundas e diverti-me bastante.

Porque todos sabemos que o melhor fica para o fim, agradeço aos meus grandes amigos de SAC, principalmente ao Costa, ao Telmo e ao Veloso por tudo que vivemos juntos, pelas lutas que nos tornaram mais fortes, pelo nosso desejo de querer sempre mais e por nunca desistirmos dos nossos sonhos. Com o passar do tempo vamos crescer, até que um dia chegaremos ao limite (da verdade).

Um grande obrigado a todos!

Aos bons momentos.

Resumo

A grande quantidade de informação que circula nas redes sociais pode ser útil e até necessária para descobrir novas ameaças de segurança. A análise manual destes dados não é prática devido ao seu elevado volume, portanto novos métodos de filtrar e classificar este fluxo de informação devem ser estudados. Pretendemos com este trabalho produzir uma aplicação capaz de detetar ameaças à segurança informática a partir da rede social Twitter de forma automática, precisa e eficiente. Para isso recorremos a técnicas de aprendizagem automática implementadas em sistemas distribuídos para alcançar um bom desempenho para a grande quantidade de dados que precisa de ser processada. Esta aplicação será capaz de extrair informação do Twitter e classificá-la de forma correta para ser analisada, dispensando intervenção manual extensiva por parte do analista. Esperamos então explorar novas soluções na área da segurança informática, necessárias numa realidade onde as organizações dependem da segurança dos seus sistemas informáticos para continuar a desempenhar as suas funções.

Palavras-chave: Machine learning, Distributed systems, Information Security, Data mining

Abstract

Social networks contain information that can be used to detect new security threats. Since manual analysis of this information is not practical due to the high volume of data, new techniques of processing this information flow must be studied. Our objective is the creation of an application capable of automatically detecting information and computer security threats from Twitter. To do so, we use machine learning algorithms implemented in large distributed systems that can scale on demand, in order to reach high throughput even as the data flow grows. This application will be able to extract information from Twitter and classify it correctly, so it can be later analysed by a security expert, avoiding the burden of manual analysis in a set of raw data. With this project we hope to explore new paths in the area of information security and make contributions to further research, much needed in a world where organizations depend on the security of their information systems to develop their activities.

Keywords: Machine learning, Distributed systems, Information Security, Data mining

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A necessidade das organizações em garantir a segurança das suas infraestruturas informáticas cresce à medida que vão dependendo destas para oferecer os seus serviços. Por exemplo, as empresas que possibilitam a compra de produtos através de uma página *web* são gravemente afetadas na sequência de um ataque à disponibilidade dos servidores ou bases de dados. Se os clientes ficam impedidos de completar as suas transações, para além de perder o lucro das compras também perde a reputação e o respeito por parte dos clientes. Outras organizações, maioritariamente governamentais, aderem à tecnologia para evitar os constrangimentos e dificuldades que podem ocorrer no atendimento presencial, como é o caso da Segurança Social ou o Portal das Finanças.

Em qualquer um destes casos, quando possível, deve-se evitar interrupções de serviços causadas por vulnerabilidades contidas nos sistemas informáticos. Com o decorrer do tempo, temos assistido a fugas de informação sensível de cidadãos de todo o mundo devido a sistemas com vulnerabilidades ou mal configurados e a complexidade crescente dos ataques realizados. Grandes empresas, com volumes de negócios na ordem dos milhares de milhões de dólares são vítimas destes ataques que afetam gravemente a confiança dos consumidores. Portanto, a segurança da informação deixou de ser um luxo e passou a ser uma necessidade, pois o risco da ocorrência de algum incidente aumenta ao longo do tempo e está diretamente relacionado com a falta de sistemas, procedimentos ou recursos humanos que ajudam a prevenção, mitigação e recuperação dos ataques feitos aos sistemas de informação. Sendo o risco conhecido e as suas consequências analisadas, é do maior interesse destas grandes empresas o investimento em ferramentas e equipas especializadas na área da segurança.

Os especialistas na área de segurança, como administradores de sistemas, consultores, analistas e equipas de segurança têm as tarefas de garantir a boa configuração dos sistemas e rede da organização, monitorizar todos os componentes de forma a detetar algum indício de ataques e efetuar testes de penetração para testar a resiliência da infraestrutura de segurança. Mas também têm a tarefa de detetar com a maior rapidez as

vulnerabilidades dos seus sistemas para que possam ser elaboradas as soluções que visam a continuidade dos serviços. Esta tarefa de prevenção requer muita pesquisa por parte do especialista. Terá de consultar várias fontes de informação relevante como *websites* de bases de dados de vulnerabilidades, *blogs* ou fóruns e consolidar os resultados para aplicar os controlos certos.

Na atualidade, as redes sociais desempenham um papel muito importante na propagação de informação. Com maior regularidade, especialistas de segurança começam a usar as redes sociais, aproveitando-se da sua abrangência para partilhar e recolher informação com outros utilizadores ligados a esta área. As próprias empresas que oferecem serviços de segurança da informação, como a *Symantec* e *Kaspersky Lab* têm páginas e perfis nas redes sociais como forma de expandir os seus negócios, mas também de partilhar boas práticas de segurança e vulnerabilidades recentemente encontradas. Para além disso, utilizadores maliciosos podem usar as redes sociais para partilhar informação sobre ameaças a sistemas informáticos específicos. Esta fonte de informação é então interessante pois pode trazer vantagens para quem a extrair de forma eficiente.

1.1 Motivação

O problema que se pretende combater com este trabalho é a falta de mecanismos que permitam a um analista de segurança analisar um fluxo de dados criado a partir das redes sociais. Com grande probabilidade de conter informação importante para a descoberta de ameaças de segurança, este fluxo não pode ser ignorado quando o objetivo é abranger todas as fontes de informação disponíveis. Por exemplo, os SOC (*Security Operation Centers*) oferecem deteção e resposta a incidentes de segurança através da correlação de informação que é monitorizada. Seria um benefício para qualquer SOC introduzir um novo sensor, que tem como fonte de informação as redes sociais e como objetivo descobrir ameaças de segurança. Mas isso inclui uma série de obstáculos, pois a informação que se encontra nas redes sociais é muito vasta e o sensor só deve capturar um subconjunto muito pequeno de informação relevante.

Por isso este trabalho tem possibilidade de causar impacto na área da segurança informática ao explorar lacunas existentes na realidade das organizações, nomeadamente nas funções desempenhadas pelos profissionais de segurança. Ao adicionar de forma simplista mais uma fonte de informação, a capacidade de resposta face a novas ameaças é melhorada. Pretendemos fazer um contributo fundamental ao tirar o maior proveito possível das tecnologias existentes para garantir que as ameaças descobertas sejam apresentadas em tempo útil e que ofereçam um elevado grau de confiança.

1.2 Objetivos

O objetivo deste trabalho é a criação e avaliação de uma aplicação capaz de classificar a informação extraída da rede social Twitter, com o propósito de descobrir novas ameaças de segurança que possam afetar a infraestrutura informática de uma organização. Devido à grande quantidade de dados do Twitter, requer-se que esta solução possa ser aplicada de forma distribuída para atingir elevados níveis de processamento e que seja facilmente escalável, consoante o volume de dados. Para conseguir processar uma quantidade tão grande de dados e devido à complexidade da tarefa de classificação de texto, recorremos a técnicas de aprendizagem automática. No final, a nossa aplicação tem de ser capaz de produzir resultados da descoberta de ameaças de forma precisa para que possa ser viável a análise desta informação por parte de um humano.

Propomos que a nossa aplicação seja um módulo adicionado a sistemas já existentes, pois percebemos que recorrer unicamente a informação extraída das redes sociais seria insuficiente. Como extensão de um sistema já implementado numa organização, deve-se ter em conta que os analistas já têm as suas mãos cheias de trabalho e, portanto, a diminuição do tempo de processamento e a redução aos resultados relevantes obtidos pela nossa aplicação torna-se prioritária. O objetivo deste trabalho está então descrito de forma simples na figura 1.

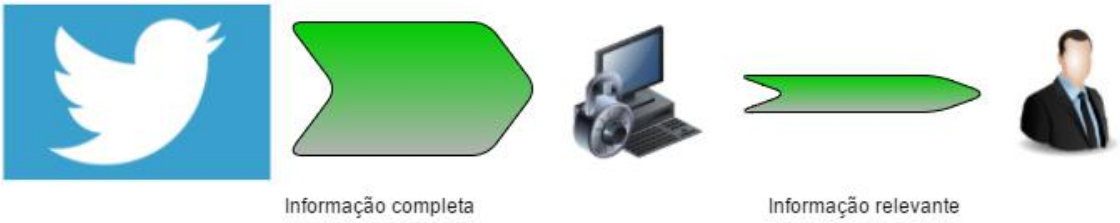

Figura 1: Vista geral da nossa proposta.

Ao usar dados do Twitter, estamos a aproveitar uma fonte de informação pública, de fácil acesso, muito volumosa e isenta de custos. Também nos queremos aproveitar da estrutura da rede social para facilitar o processo de recolha de dados. Através desta fonte de informação pretendemos obter uma maior capacidade de resposta face à crescente vaga de vulnerabilidades e ameaças que são descobertas.

1.3 Contribuições

A combinação de técnicas de aprendizagem automática, recolha de dados nas redes sociais e a intuição humana de profissionais de segurança informática permitiu-nos elaborar esta ferramenta, que é capaz de classificar dados de acordo com uma série de requisitos elaborados na sua produção.

As principais contribuições deste trabalho são:

- O uso de máquinas de vetores de suporte (MVS) (*Support Vector Machines*) [1] na descoberta de ameaças de segurança publicadas na rede social Twitter.
- A utilização de filtros baseados em palavras-chave para reduzir o volume de dados usados na criação do classificador e na classificação de dados desconhecidos.
- A divisão da infraestrutura de TI de uma organização com o objetivo de simplificar a análise dos resultados e melhorar o desempenho dos classificadores.
- O desenvolvimento da solução usando uma biblioteca que permite a sua execução num ambiente distribuído.
- A avaliação do trabalho desenvolvido recorrendo a um caso de uso que permite verificar a sua viabilidade.

Contudo, reconhecemos que o trabalho aqui proposto pode ser alvo de melhorias, nomeadamente na atualização dos modelos de aprendizagem automática que podem perder as suas capacidades ao longo do tempo, sendo necessária uma fase de intervenção humana adicional para refinar o desempenho dos modelos. Também podem ser integradas técnicas baseadas em conhecimento factual para reduzir ainda mais a intervenção humana e portanto, melhorar a qualidade do trabalho.

1.4 Publicações

O trabalho realizado nesta dissertação permitiu a publicação de um artigo completo na conferência INForum 2016, sobre a sessão "Segurança de Sistemas de Computadores e Comunicações" [2].

1.5 Estrutura do Documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Neste capítulo iremos introduzir alguns conceitos teóricos e ferramentas usadas para produzir a nossa solução, nomeadamente as características das fontes de informação abertas, a representação e transformação dos dados nos formatos necessários, a técnica de aprendizagem automática usada e finalmente a ferramenta usada para programar a nossa solução. Também está incluído no final deste capítulo o trabalho relacionado.
- Capítulo 3 – Neste capítulo iremos explicar o problema que a nossa solução pretende resolver, introduzindo em detalhe a nossa arquitetura e todas as decisões e abordagens importantes que tomámos e foram cruciais para chegar ao produto final.

- Capítulo 4 – Demonstramos as experiências efetuadas através de um caso de uso representativo, fazendo um enquadramento da combinação de todas as ferramentas e abordagens usadas e discutimos os vários resultados obtidos com base num extenso processo de avaliação.
- Capítulo 5 – Apresentamos as conclusões retiradas da elaboração do nosso trabalho através de uma curta análise, incluindo no final extensões que podem ser adicionadas em trabalhos futuros para melhorar o desempenho do sistema proposto.

Capítulo 2

Ferramentas e Trabalho Relacionado

Neste capítulo vamos descrever os vários componentes do trabalho realizado. Começamos por explicar o interesse que há em aproveitar fontes de informação abertas e de que forma usamos uma dessas fontes (Twitter) para recolher informação. De seguida, incluímos duas técnicas de extração de características usadas para representar os dados e também o algoritmo de aprendizagem automática usado para processar os dados. Discutimos a escolha da ferramenta usada para implementar todos estes componentes e finalmente refletimos sobre o trabalho realizado nas áreas de aprendizagem automática em redes sociais, descoberta e previsão de ameaças em redes sociais e *data mining*.

2.1 OSINT

Open Source Intelligence (OSINT) [3] é a inteligência extraída a partir de informação pública, normalmente em formato eletrónico ou impresso, que pode ser obtida legalmente e eticamente através de fontes públicas como a Internet, rádio, televisão e jornais. Está mais distribuída e difundida pela sociedade, em vez de estar contida num grupo fechado de utilizadores dentro da sociedade.

Devido às fontes de informação serem públicas, o problema não é a sua disponibilidade mas sim a capacidade de localizar a informação relevante e consolidá-la de modo a ser aproveitada. Também é uma forma de poupar recursos pois esta informação pode ser obtida a partir de fontes comerciais a um preço baixo ou gratuitamente e em menos tempo e ainda permite uma melhor alocação de recursos para ter o melhor proveito.

Historicamente, a OSINT é usada a nível militar e complementa as vantagens oferecidas pelos métodos tradicionais de obter inteligência. Em operações militares, existem casos em que OSINT é a única fonte de informação capaz de oferecer resultados rapidamente. Permite resolver problemas de confiança entre agências de nações diferentes quando estas são obrigadas a cooperar, e também garante que os agentes a tratar de OSINT não vão trair as suas agências de inteligência pois o valor da OSINT é baixo (pode ser adquirido publicamente).

Um exemplo da importância da OSINT é o seu uso para antecipar ameaças de terrorismo [4], através da busca de informação em *blogs* e *fóruns*. A Internet é considerada o meio de propagação de ideologias terroristas mais importante e portanto as agências de segurança dos diversos países encontraram utilidade em explorar OSINT.

Contudo, existem alguns problemas e desafios nas abordagens ao tratamento da OSINT. A quantidade de informação disponível é imensa e as aptidões dos profissionais que analisam estes fluxos de informação tem de aumentar também. Existe também o problema da redundância da informação, em que uma fonte publica informação que depois é capturada e re-publicada por outras entidades. Ocorre muito no jornalismo, onde uma agência publica uma noticia que de seguida é aproveitada por dezenas de outras agências, cada uma relatando a noticia de maneiras diferentes. Esta redundância pode confundir os profissionais da OSINT.

O desafio principal da OSINT e que pretendemos abordar no nosso trabalho, é saber que dados são úteis para resolver um determinado problema. Também temos de ter em conta a quantidade de dados necessária e o tempo necessário para recolher os dados. Os dados OSINT, como são de livre acesso e podem ser publicados por qualquer pessoa, vão ser alvo de alguma imparcialidade, incoerência ou irrelevância. Ou seja, podem ser tão prejudiciais como são benéficos, ao desinformar os analistas que tratam estes dados.

2.2 Twitter

Twitter [5] é uma rede social que permite aos seus utilizadores publicar pequenos textos de 140 caracteres mas também conteúdo multimédia, como vídeos ou imagens. A interligação entre os utilizadores é feita através do conceito de "seguidor", em que utilizadores seguem outros que façam publicações dentro de determinados tópicos de interesse. Quando um utilizador seguido publica algum *tweet*, irá aparecer na *timeline* de todos os utilizadores que o estiverem a seguir. Também tem como característica as *hashtags*, que são palavras precedidas pelo caracter '#' que permitem aos utilizadores encontrar com maior facilidade tópicos do seu interesse. Os utilizadores podem fazer pesquisas por *hashtags*, que apresentarão todas as publicações que contêm essas *hashtags*.

O número de utilizadores do Twitter tem vindo a aumentar, como se pode ver na figura 2, devido à sua simplicidade e capacidade de poder partilhar informação com muito pouco esforço. Isto faz do Twitter uma excelente rede social para estudar a resolução de determinados problemas que envolvem *data mining*, quer seja para descobrir a opinião geral do público perante um produto ou analisar a interação em tempo real dos utilizadores presentes em catástrofes naturais.

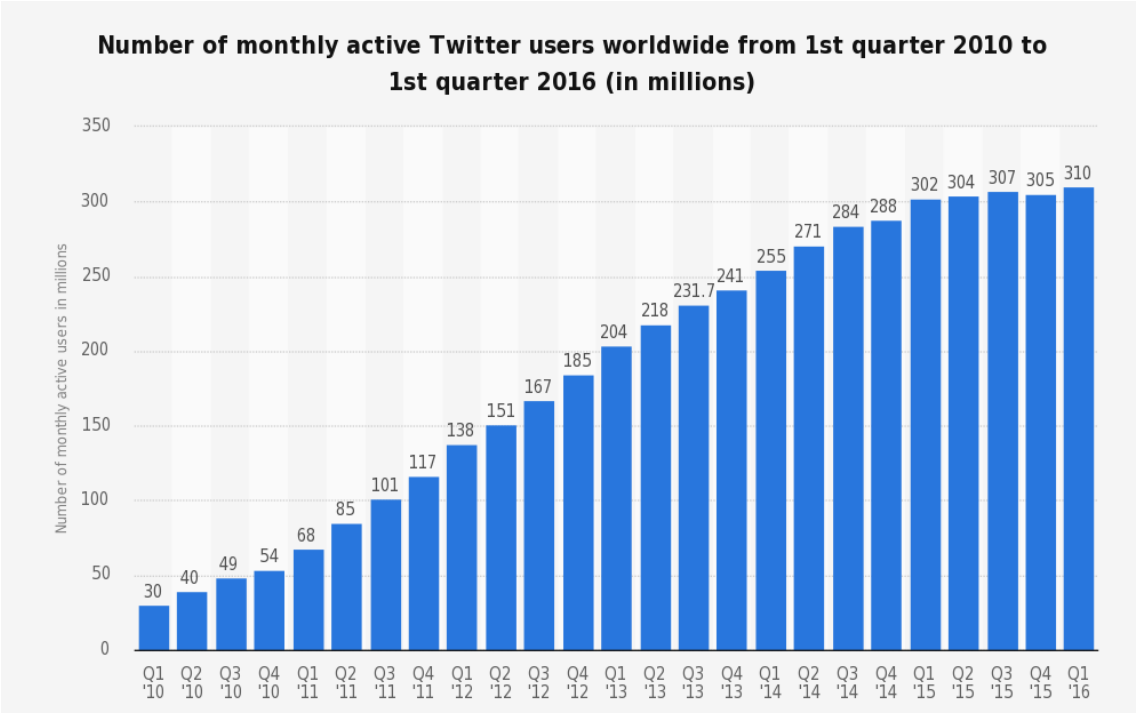

Figura 2: Número de utilizadores do Twitter ativos entre 2010 e 2016 [49].

2.3 Representação de Dados

A construção de um classificador pode ser vista como o problema de encontrar uma relação entre observações e resultados usando um conjunto de dados como exemplo, de forma a minimizar algum critério de performance [6]. Neste trabalho o objetivo é classificar corretamente *tweets* de acordo com algumas das suas características. Como os *tweets* são mensagens de texto e os algoritmos de aprendizagem automática são numéricos, torna-se necessário encontrar uma representação numérica (vetorial) para cada *tweet*. De uma forma genérica este problema designa-se extração de características (*feature extraction*) [6]. Numa das abordagens usadas os *tweets* são transformados respeitando uma representação treinada a partir de um conjunto de dados complexo enquanto na outra abordagem é usado um modelo *bag-of-words*, em que uma frase é representada por um multiconjunto composto pelas suas palavras, que pode ser facilmente convertido em um vetor.

2.3.1 TF-IDF

Nas tarefas de classificação de texto, as palavras mais comuns são aquelas que ajudam a formar frases e ideias, mas que por si só não trazem nenhum significado, enquanto as palavras mais raras têm maior probabilidade de indicar o significado de uma frase. Mas entre as palavras que aparecem com menos frequência, existem aquelas que

também trazem pouco significado. Então é necessário saber distinguir que palavras se devem realçar e que palavras não devem ter tanta importância. TF-IDF (*Term Frequency – Inverse Document Frequency*) [7] permite-nos ultrapassar essa dificuldade.

Supondo que temos um conjunto de *N* documentos (que podem ser frases ou *tweets*), definimos como f_{ij} a frequência de um termo *i* no documento *j* e calculamos a frequência (normalizada) do termo (TF_{ij}) da seguinte forma:

$$TF_{ij} = \frac{f_{ij}}{\max\limits_{k} f_{kj}}$$

Supondo que n_i é o número de documentos em que o termo *i* aparece, então:

$$IDF_i = \log_2\left(\frac{N}{n_i}\right)$$

Finalmente, podemos calcular a pontuação TF-IDF para cada termo *i* no documento *j* da seguinte forma:

$$TF\text{-}IDF = TF_{ij} \times IDF_{ij}$$

Através desta técnica, conseguimos distinguir que palavras são mais importantes para caracterizar uma frase com base na sua pontuação.

2.3.2 Word2Vec

Word2Vec [8] é um projeto *open-source* que permite a criação de vetores numéricos de alta qualidade a partir de conjuntos de dados grandes (na ordem dos milhares de milhões de palavras). O objetivo é uma representação vetorial que reflete a semelhança entre várias palavras, não só lexicograficamente mas também sobre os seus significados.

No trabalho apresentado em [9], são propostas operações entre vetores que demonstram o potencial desta abordagem. Um exemplo dado é a seguinte operação:

vec(“Madrid”) - vec(“Spain”) + vec(“France”)

O vetor resultante desta operação está mais próximo de vec(“Paris”) do que qualquer outro vetor. Esta representação distribuída de palavras num espaço vetorial é benéfica para aumentar o desempenho em tarefas de processamento de linguagem natural, ao agrupar palavras com significado semelhante.

Os criadores deste projeto tiveram como objetivo reduzir a complexidade computacional, então evitaram usar redes neuronais e criaram os seus próprios modelos. O primeiro é o *Continuous Bag-of-Words*, em que para obter a representação de uma palavra **p**, usam-se as quatro palavras que antecedem **p** e as quatro palavras que sucedem **p** no texto usado para treinar o modelo. O segundo modelo é o *Continuous Skip-gram Model*, que em vez de prever a palavra atual **p** com base nas palavras em redor, a palavra

p contribui para obter a representação das palavras que antecedem e sucedem **p**. Mais tarde, apresentaram uma extensão a este modelo. Um dos grandes problemas nas tarefas de processamento de dados, como foi descrito na secção anterior, são as palavras frequentes que oferecem menos (ou quase nenhuma) informação que as palavras que ocorrem menos vezes. Então, foi introduzida uma função que descarta as palavras mais frequentes, com base num limite previamente definido [10]. Com esta redução de texto, o processo de treino torna-se mais rápido e a precisão dos vetores das palavras menos frequentes é maior.

Na sua implementação, um modelo Word2Vec constrói um vocabulário a partir de um conjunto de dados grande e aprende a representação vetorial das palavras. A qualidade da representação está dependente do conjunto de dados usado para criar o vocabulário.

Embora seja um projeto muito jovem, decidimos usá-lo paralelamente ao TF-IDF porque acreditamos que para o nosso projeto, seria benéfico que as frases que contenham palavras relacionadas com segurança informática (ex: *vulnerability*, *threat*, *malware*) fossem representadas por vetores muito próximos no espaço vetorial.

2.4 Máquinas de Vetores de Suporte

Tal como descrito por Simon Haykin [1], máquinas de vetores de suporte são algoritmos de aprendizagem automática que possibilitam efetuar sobre um conjunto de dados uma classificação binária. Para treinar o modelo, é necessário um conjunto de dados já categorizados (positivos e negativos). Com estes dados, a máquina de vetores de suporte constrói um hiperplano como decisor/classificador de forma a maximizar a margem de separação entre os exemplos positivos e negativos, tal como se pode ver na figura 3.

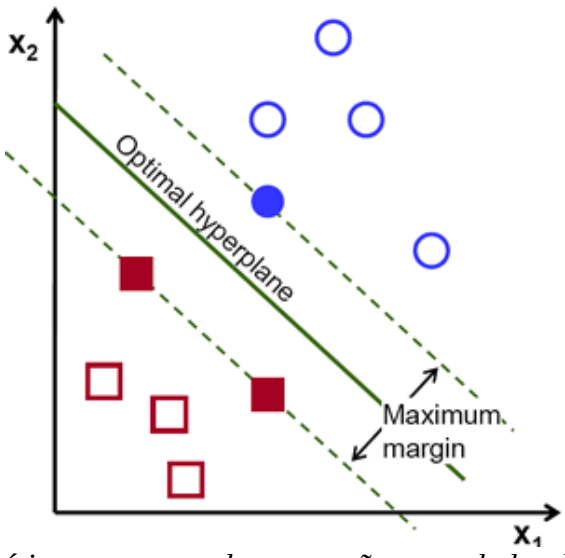

Figura 3: Hiperplano ótimo e margem de separação entre dados das classes quadrado e circulo [48].

Os vetores de suporte, são pontos retirados dos exemplos de treino, sendo os que estão mais próximos do hiperplano. Os restantes não são usados pela máquina de vetores de suporte após a sua determinação.

O método de escolher o hiperplano ótimo difere consoante os dados de entrada. Estes dados podem ser linearmente separáveis ou não-linearmente separáveis. No caso dos dados linearmente separáveis, a equação da superfície de decisão que constrói o hiperplano é:

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0$$

onde $\boldsymbol{w}^T$ são os pesos, $\boldsymbol{x}$ é o vetor de entrada e b é o viés.

Dado um conjunto de vetores de entrada, determinar a máquina de vetores de suporte consiste em encontrar os pesos e o viés que permitem definir o hiperplano em que a margem de separação, ou seja, a distância entre o hiperplano e os pontos de entrada mais próximos seja maximizada. Da otimização resulta que os vetores de suporte são os pontos mais próximos do hiperplano e que portanto, são mais difíceis de classificar.

Determinar o hiperplano ótimo significa resolver um problema de otimização convexa com restrições.

Sendo

$$\mathrm{t} = \{\boldsymbol{x}_i, d_i\}_{i=1}^{N}$$

os exemplos de treino, onde $\boldsymbol{x}$ é um vetor de entrada e d é a sua classificação (-1 ou 1), é necessário encontrar os valores de $\boldsymbol{w}^T$ e b que satisfaçam a seguinte restrição:

$$d_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \quad \text{para } i = 1,2,\dots,\mathrm{N}$$

e cujo vetor de peso w minimize a função de custo:

$$\varphi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

O grande desafio na construção de uma máquina de vetores de suporte é resolver o problema de otimização. Inicialmente, resolveu-se este problema usando o método dos multiplicadores de *Lagrange*. Estas variáveis auxiliares, denominadas por α_i, permitem a construção da seguinte função de custo:

$$J(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i[d_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1]$$

A solução do problema de otimização passa por encontrar o ponto que torna as raízes da equação $J(\boldsymbol{w}, b, \alpha)$ reais mas com sinais opostos. Podem existir vários destes pontos, portanto é necessário encontrar aquele que é minimizado em relação a $\boldsymbol{w}$ e b mas que é

maximizado em relação a α. A descrição completa bem como as origens do método podem ser consultados em [1,11].

No caso dos dados de entrada não serem linearmente separáveis, usa-se outra abordagem. Nestes casos, não é possível construir um hiperplano sem que haja erros de classificação. Então o objetivo é encontrar um hiperplano que minimize a probabilidade de haver erros de classificação.

As máquinas de vetores de suporte são uma das soluções mais usadas em problemas de classificação [12] devido à sua capacidade de apresentar bons resultados. Contudo, quanto maior for a quantidade dos exemplos de treino, maior são os requisitos computacionais e de armazenamento, devido a complexidade de resolver as otimizações quadráticas usadas no algoritmo de treino. Isto faz com que as tarefas que tenham grandes quantidades de dados não sejam adequadas às máquinas de vetores de suporte, sendo esta dificuldade ultrapassada através da implementação distribuída deste algoritmo.

2.5 Apache Spark

Um dos grandes desafios deste trabalho foi encontrar a ferramenta certa para desenvolver a aplicação. Bibliotecas como o Scikit-learn [13] podem ser usadas para aplicar algoritmos de aprendizagem automática, contudo não se enquadram com o objetivo de processamento em grande escala. Portanto, o estudo das ferramentas teve em conta certos critérios, como a facilidade da criação de aplicações, o número de algoritmos de aprendizagem automática implementados e o desempenho em *cluster*.

Apache Mahout [14] é um projeto que tem como objetivo oferecer um ambiente que permita a criação de aplicações de aprendizagem automática de forma escalável. Para isso, as versões iniciais usavam o modelo de programação MapReduce [15] e o ambiente de execução do Hadoop [16] como base, que permitia o processamento de grandes quantidades de dados de forma distribuída e tolerante a faltas. Contudo, visto que as operações do Hadoop efetuam muitas escritas no disco, é natural que a performance das aplicações que o usam seja afetada. Este era um dos pontos fracos do Mahout, que em tarefas que precisem um elevado número de iterações, tais como as necessárias para aplicações de aprendizagem automática, acaba por se tornar ineficiente. Então, no projeto Mahout, o Hadoop foi substituído pelo Spark

Apache Spark [17], tal como o Hadoop, é uma ferramenta para a criação de aplicações que envolvem o processamento de uma grande quantidade de dados. Foi criado pela necessidade de aumentar a velocidade deste tipo de aplicações, que não podia ser alcançada através do MapReduce. Com o Spark, introduziu-se o conceito de *Resilient Distributed Datasets* (RDD) [18] que suportam as aplicações que reutilizam os resultados intermédios em cada iteração das computações. Os RDD são um conjunto de partições

que podem ser armazenadas em memória volátil ou em memória persistente e que só podem ser criadas a partir de transformações. Cada RDD é capaz de saber como foi derivado a partir do seu histórico de transformações. Este histórico de transformações serve para alcançar tolerância a faltas, pois se uma partição do RDD falhar, é possível voltar a recriá-la observando o histórico e aplicando o conjunto de transformações aos dados iniciais da partição (que estão em disco).

O desempenho dos RDD é bom se o conjunto de dados a serem alvo das operações estiver todo em memória. Quando isto não acontece, então as partições podem ser armazenadas em disco.

Os RDD devem ser usados em aplicações que usam a mesma operação a todos os elementos do conjunto de dados. Desta forma, os RDD conseguem registar facilmente as transformações que foram aplicadas às partições usando pouco espaço em disco. Sendo assim, visto que maior parte dos algoritmos de aprendizagem automática aplicam várias operações iterativamente, podemos afirmar que estes algoritmos são executados muito melhor quando os dados se encontram em memória.

Mas o desempenho é só mais uma das múltiplas vantagens que o Spark tem em relação às outras ferramentas. A configuração pode ser feita de forma simples, instalando uma versão compilada do Spark em cada uma das máquinas e executando os comandos que fazem a ligação com a máquina Mestre (*Driver Program*). Quando todas as máquinas estiverem a comunicar, basta submeter uma aplicação e a computação é distribuída no *cluster*. A figura 4 mostra a arquitetura do Spark. O *Driver Program* contém e coordena cada aplicação, atribuindo tarefas aos *Worker Nodes*, que atualizam e armazenam os dados da aplicação. O *Cluster Manager* é o componente responsável pela comunicação entre o *Driver Program* e os *Worker Nodes*.

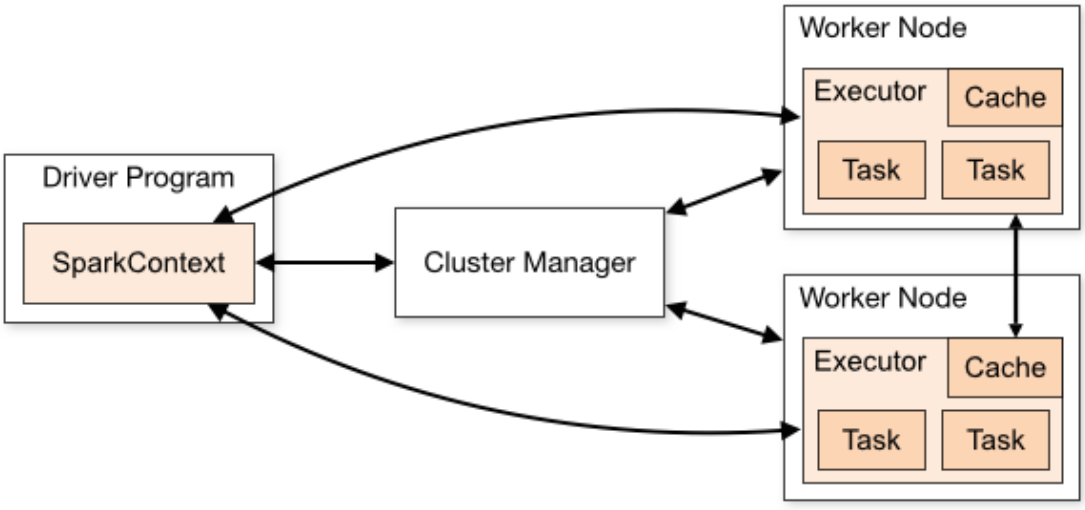

Figura 4: Arquitetura do Spark em ambiente de cluster [47].

Além disso, o Apache Spark conta com uma comunidade bastante ativa de utilizadores e programadores que mantém o projeto atualizado [8]. Face a estas vantagens todas, decidimos escolher a biblioteca MLlib [19], que já vem integrada no Spark e é responsável pela implementação de vários algoritmos de aprendizagem automática. Neste trabalho iremos usar a implementação das máquinas de vetores de suporte lineares do MLLib que usa o *Stochastic Gradient Descent* [20] como algoritmo de treino.

2.6 Trabalho Relacionado

2.6.1 OSINT e Aprendizagem Automática em Segurança Informática

O uso de ferramentas para descobrir ameaças de segurança através das redes sociais ainda é uma área pouco explorada. Ritter et al. [21] recorre a uma abordagem fracamente supervisionada para extrair eventos de segurança. O analista de segurança neste caso apenas precisa de fornecer uma pequena amostra de eventos de segurança relevantes. A partir destes eventos são extraídos *tweets* que mencionam que um ataque ocorreu à entidade na data especificada, sendo estes *tweets* usados para treinar o modelo que vai conseguir categorizar eventos de segurança. Esta solução apresenta uma forma simples de criar um classificador que não necessita de apresentar exemplos de eventos não relevantes, facilitando assim o trabalho do analista.

Sabottke et al. [22] usou informação do Twitter para detetar vulnerabilidades que efetivamente são "exploráveis", através de um classificador que correlaciona uma vulnerabilidade divulgada por um feed de segurança fiável com posts a seu respeito no Twitter. O autor discute as oportunidades de detetar estas vulnerabilidades através de informação proveniente de fontes públicas mas que não se encontra nas bases de dados de vulnerabilidades. Um dos seus grandes desafios é combater o ruído provocado pelos utilizadores que mencionam as vulnerabilidades mas não dão informação nenhuma sobre como as explorar. Para além disso, também considerou o caso de os dados do Twitter serem maliciosamente colocados para produzir erros no seu classificador.

Rodrigues [23] encaminhou um fluxo de dados OSINT (*Open Source Intelligence*), mais precisamente do Twitter, para a plataforma HP ArcSight de modo a tirar proveito de mais uma fonte de informação para detetar vulnerabilidades. Tal como Sabottke et al., a sua solução classificou os *tweets* com base na ocorrência e frequência de palavras, sendo que apenas os *tweets* mais importantes eram transferidos para a uma plataforma SIEM. Esta solução precisa apenas de um conjunto de palavras-chave que irão ser utilizadas para pesquisar *tweets*, que serão então ordenados consoante a sua pontuação. Esta pontuação, para além de depender da frequência das palavras-chave, também depende de um mecanismo de feedback em que os analistas podem atribuir pesos maiores a certas

palavras-chave. Deste modo, os analistas conseguem exprimir o seu interesse na descoberta de informação mais específica.

Finalmente, uma solução mais recente foi desenvolvida por Veeramachaneni et al. [24] que construiu um sistema (AI2) que analisa *logs* usando algoritmos de aprendizagem automática juntamente com um mecanismo onde a intuição do analista é considerada. Tenta combater a dependência dos analistas para investigar ataques ao criar modelos de forma automática que, quando executados sobre novos dados, determinam resultados semelhantes aos deduzidos por analistas humanos. A intervenção humana tem o papel de modificar constantemente os modelos de classificação, para que estes se mantenham atualizados.

2.6.2 Processamento Automático de *Tweets*

O uso das redes sociais por milhões de utilizadores permite que estes partilhem as suas opiniões sobre os mais diversos assuntos. Uma das áreas com maior foco de investigação é a análise de sentimentos nas publicações das redes sociais. Esta área é estratégica para as grandes empresas que pretendem saber a opinião dos consumidores sobre determinados produtos enquanto que para os consumidores, pode contribuir para a decisão de efetuar uma compra. Agarwal et al. [25] classificam *tweets* em positivos e negativos, com base no sentimento que o utilizador quer exprimir. Isto é feito através de um dicionário que consegue associar palavras, *emoticons* e certos acrónimos a uma *tag* que representa o nível de sentimento, que vai de extremamente negativo a extremamente positivo. Após substituírem os termos pelas *tags*, podem aplicar máquinas de vetores de suporte nos *tweets* para produzir os modelos. Coletta et al. [26] mostram que a combinação de um classificador com um módulo de *clustering* é capaz de produzir melhores resultados que apenas um classificador na análise de sentimentos. Jiang et al. [27] e Go et al. [28] estudam uma abordagem que consiste em classificar *tweets* de acordo com o seu sentimento exprimido perante um determinado assunto. No primeiro, combatem a dificuldade de classificar *tweets* muito pouco específicos através da correlação de *tweets* publicados que mencionam o mesmo assunto enquanto que no segundo fazem um esforço para reduzir a intervenção manual da classificação de dados positivos e negativos ao definir um conjunto de dados constituído apenas por *tweets* que tenham *emoticons* alegres, tais como :), :D, como sendo *tweets* que exprimem um sentimento positivo e *tweets* que tenham *emoticons* tristes como sendo *tweets* que exprimem sentimentos negativos. Wang et al. [29] não estudam o sentimento de um *tweet* perante um tópico mas sim o sentimento atribuído a *hashtags*, através da análise das publicações que contém a *hashtag*, da correlação entre *hashtags* e do significado literal da *hashtag*. Outros investigadores estudam os sentimentos expressos nas mensagens do Twitter para tentar prever resultados eleitorais, para saber a opinião dos utilizadores em

relação aos políticos a participar numa campanha eleitoral e para prever as intenções de votos de um país inteiro (Irlanda) durante uma campanha eleitoral [30] [31] [32].

Também pode ser útil observar as redes sociais para a deteção de eventos incomuns como catástrofes naturais, devido à elevada afluência de utilizadores presentes que relatam o acontecimento. Sakaki et al. [33] explora a interação em tempo real dos utilizadores do Twitter para detetar terramotos, Aramaki et al. [34] estuda o uso de MVS para detetar epidemias de influenza através de publicações no Twitter, encontrando dificuldades em ter um bom desempenho quando a época era propícia a gripes e portanto, muito falada nos noticiários. Burnap et al. [35] tentam combater o problema da propagação de informação relacionada com suicídios a circular no Twitter ao usar MVS para classificar publicações que tenham este tipo de conteúdo. O problema principal é conseguirem distinguir informação que incentiva à prática do suicídio e informação que relata casos de suicídio ou que contribui para a prevenção do suicídio. Zangerle et al. [36] investigam a ação dos utilizadores do Twitter ao saberem que as suas contas foram comprometidas através da análise aos seus *tweets*. Neste caso a dificuldade passa por diferenciar mensagens que relatam que a própria conta foi comprometida de mensagens que mencionam contas de outras pessoas.

Na categoziação de documentos, Lee et al. [37] propõe um método de atribuição de categorias a documentos usando diversas técnicas de aprendizagem automática e um sistema de votação. Os documentos são classificados com base nos resultados de três classificadores independentes, sendo a classificação final a maioria dos resultados dos classificadores. Lee et al. [38] atribui categorias aos *tweets* que estão incluídos nos tópicos mais populares do Twitter, usando uma abordagem baseada na própria rede social, através da relação entre utilizadores, em vez de usarem apenas o texto das publicações. Dilrukshi et al. [39] classifica notícias publicadas no Twitter em grupos diferentes de modo a que os utilizadores consigam identificar o grupo de notícias mais popular. Rao et al. [40] pretende descobrir automaticamente características dos utilizadores como a idade, género, orientação politica e região a partir das publicações dos utilizadores, estrutura da rede social e comunicação entre utilizadores, usando diversas técnicas para cada uma destas categorias.

Outros trabalhos exploram formas de conseguir detetar publicações que se destacam pelo seu conteúdo, como Boutkan et al. [41] que pretende criar um classificador que detete frases que contenham vocabulário impróprio em holandês, ou [42] e [43] que se aproveitam de características de contas das redes sociais como o número de seguidores, o número de mensagens publicadas e o conteúdo das mensagens para descobrir *spammers.* Finalmente, Abu-Nimeh et al. [44] compara o desempenho de várias técnicas de aprendizagem automática na tarefa de detetar tentativas de *phishing* em *emails*.

2.6.3 Discussão

Podemos ver que o uso de aprendizagem automática nas redes sociais é uma excelente forma de obter informação relevante para os objetivos das organizações. O grande problema que os analistas têm é saber como procurar a informação que realmente é importante no meio de uma quantidade gigante de dados. Os algoritmos de aprendizagem automática facilitam esta tarefa, permitindo selecionar subconjuntos muito específicos que por sua vez permitem a criação de aplicações de publicidade e recomendações.

A solução proposta visa explorar áreas de estudo diferentes usadas pelos trabalhos referidos nesta secção de forma a atingir com sucesso o objetivo de detetar ameaças de segurança. É de notar que a solução difere de qualquer outro trabalho ao criar modelos de aprendizagem automática a partir do conteúdo das publicações do Twitter, que têm características diferentes das publicações referidas nos trabalhos mencionados neste capítulo ao estarem relacionadas com a área da segurança informática.

2.7 Considerações Finais

Neste capítulo começámos por descrever o papel das fontes de informação abertas e a sua importância para a descoberta de ameaças de segurança. Discutimos também a estrutura da rede social Twitter que é muito usada atualmente e, portanto, muito interessante como alvo de estudo. Tivemos em conta as dificuldades e limitações da extração de dados com as ferramentas que nos estão disponíveis e adaptámo-nos para ultrapassar essas dificuldades de modo ser possível extrair uma maior quantidade de dados. Introduzimos o problema da representação e transformação de dados, que é essencial para tirar proveito dos algoritmos de aprendizagem automática ao mesmo tempo que explicámos duas formas distintas de o fazer. Descrevemos ainda o funcionamento das máquinas de vetores de suporte através das bases matemáticas que tornam esta técnica uma das mais usadas nas tarefas de classificação de texto. Por fim, mencionamos os trabalhos relacionados com a área de estudo do nosso projeto, destacando os aspetos únicos de cada um e discutindo no final as diferenças entre o que já foi feito e o que pretendemos fazer.

Capítulo 3

Descoberta de Ameaças usando o Twitter

Neste capítulo descrevemos a nossa solução para o problema de recolher dados nas redes socais para a descoberta de ameaças a uma infraestrutura informática. Através dos conceitos e técnicas introduzidas no capítulo anterior, podemos agora detalhar a arquitetura da solução.

Começamos por descrever a implementação do módulo de recolha de dados, mencionando todos os passos e dificuldades devido às fortes restrições da API disponibilizada. De seguida, discutimos as abordagens específicas do trabalho que o diferenciam de outros na área da prospeção de dados, incluindo o tratamento de dados, a justificação do uso de filtros e múltiplos classificadores e a fase de classificação manual. Terminamos o capítulo com a explicação das decisões de implementação nas duas grandes fases que compõem o trabalho, destacando o fluxo de dados desde o início da sua execução.

3.1 Definição do Problema

Conforme apresentado no capítulo 1, o objetivo deste projeto é filtrar as informações relevantes obtidas em redes sociais (em particular, o Twitter) com mínima intervenção humana, tanto na fase de classificação manual (treino) quanto na análise dos resultados da classificação automática (operação). Para isso, temos de reduzir a quantidade de informação que tem de ser classificada pelo analista para produzir os classificadores necessários, mas nunca reduzindo ao ponto de os classificadores criados serem ineficazes.

De maneira mais precisa, após uma fase inicial de treino, o sistema tem como objetivo (1) apresentar aos analistas do SOC de uma organização um conjunto de *tweets* que seja relevante no contexto da segurança informática nessa organização (baixo nível de falsos positivos) e (2) minimizar o conjunto (idealmente vazio) de *tweets* relevantes não apresentado (baixo nível de falsos negativos).

3.2 Arquitetura

As figuras 5 e 6 apresentam a arquitetura do sistema proposto. Nestas figuras, os retângulos brancos representam os componentes do sistema, as caixas amarelas

representam a execução dos componentes e as setas verdes representam o fluxo de dados e ligação entre a execução dos componentes.

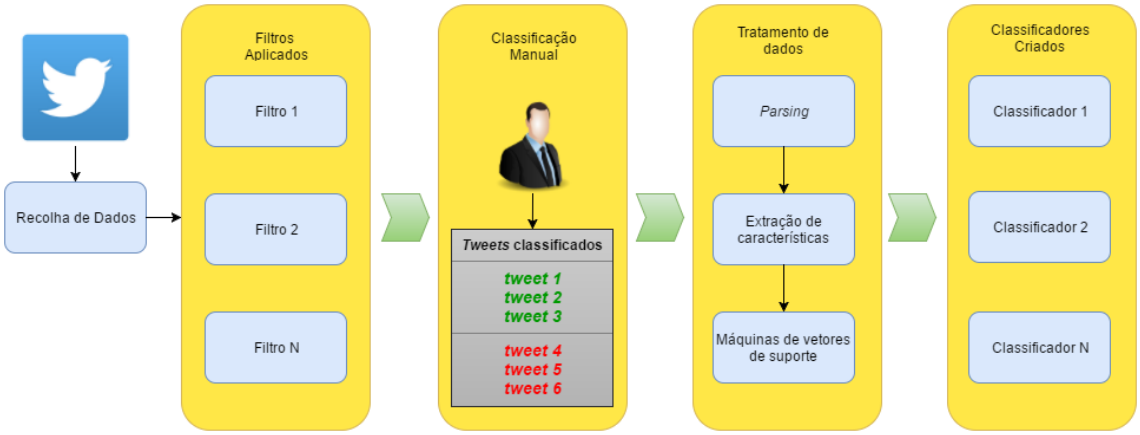

Figura 5: Arquitetura geral da aplicação em modo de treino.

O sistema tem dois modos de funcionamento: treino e operação. A divisão do sistema em dois modos deve-se à natureza das técnicas de aprendizagem automática supervisionadas, que precisam previamente de uma fase de treino com dados rotulados para conseguirem produzir os modelos capazes de prever resultados para conjuntos de dados desconhecidos. No modo de treino (ilustrado na figura 5), inicialmente, são recolhidos os dados a partir do Twitter e armazenados em ficheiros de texto. Esta recolha de dados pode ser feita periodicamente em *batch* (os dados são recolhidos em bloco a qualquer altura) ou continuamente em *streaming* (os dados são recolhidos à medida que estão disponíveis no Twitter). Os *tweets* recolhidos passam por um filtro inicial para reduzir o volume de dados a ser classificado manualmente. De seguida, o analista de segurança irá classificar manualmente os *tweets*, de modo a produzir um documento em que os *tweets* que representam ameaças de segurança à infraestrutura de TI estejam rotulados corretamente. Este documento é então alvo das transformações necessárias para reduzir a complexidade da tarefa e da extração de características, sendo então usado pelos algoritmos de aprendizagem automática para treinar um classificador capaz de distinguir *tweets* que demonstrem conter algum indício de ameaças ou vulnerabilidades aos sistemas da infraestrutura de TI considerada.

Após treinado, o classificador é alimentado com os *tweets* recolhidos e filtrados, desta vez em tempo real, e classifica-os de forma a serem analisados posteriormente pelo analista. Este fluxo de trabalho corresponde ao modo de operação da ferramenta, ilustrado na figura 6.

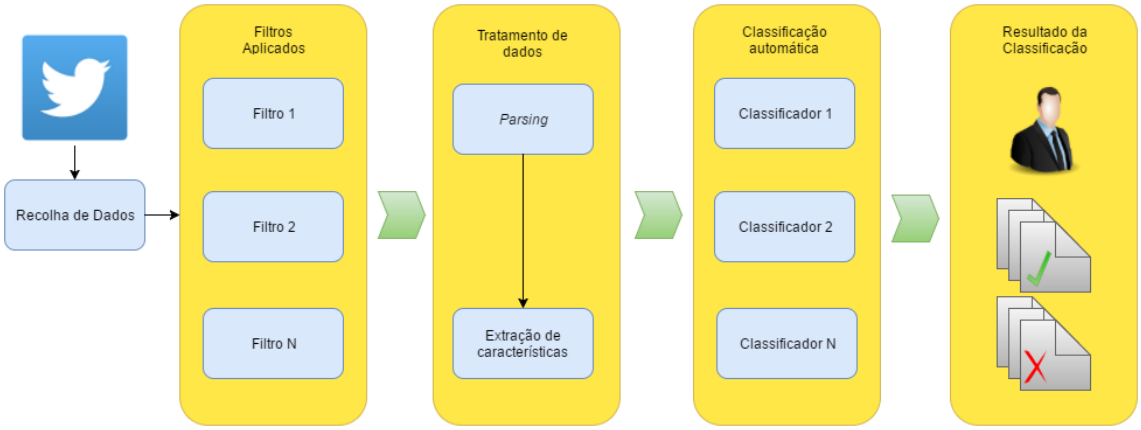

Figura 6: Arquitetura geral da aplicação em modo de operação.

Na secção seguinte iremos descrever o módulo de recolha de dados, que é o primeiro componente da arquitetura do sistema e está presente em ambos os modos.

3.3 Recolha de Dados

A recolha de dados é um dos principais componentes do sistema, pois é através dos dados recolhidos no modo de treino que serão produzidos os modelos de aprendizagem automática e serão os dados recolhidos no modo de operação os alvos da classificação dos modelos criados.

Todos os *tweets* usados neste trabalho foram extraídos através da biblioteca Tweepy [45] para a linguagem de programação Python. Esta biblioteca permite a comunicação entre o Python e a plataforma Twitter, acedendo à sua API de programador [46].

A API REST permite ao programador de leitura e escrita de dados do Twitter, permitindo por exemplo escrever novos *tweets* na sua *timeline*, listar todos os utilizadores que estão a ser seguidos ou transferir os últimos *tweets* da *timeline* de um utilizador específico. Também está disponível a API *Streaming* que disponibiliza os dados do Twitter em tempo real. No âmbito desta dissertação, a API REST foi mais apropriada porque queríamos ter acesso a *tweets* publicados no passado, para ter um *corpus* mais completo e próximo de um caso de uso real.

A funcionalidade da API REST usada foi a "**GET statuses/user_timeline**" que devolve um conjunto de *tweets* (juntamente com informação adicional, como data de publicação e identificador) recentemente publicados por um utilizador. Esta funcionalidade tem algumas limitações, como só ser possível recolher 200 *tweets* por chamada à função e um máximo de 3200 *tweets* mais recentes. Face a esta limitação, a fase de recolha de dados foi continuando ao longo do nosso projeto. Ao extrair os dados mensalmente conseguimos ultrapassar esta limitação, tendo a única preocupação de

garantir que entre extrações os utilizadores não publiquem mais de 3200 *tweets*, para não desperdiçar *tweets* que mais tarde não poderão voltar a ser recolhidos.

Vamos descrever agora os passos de execução da aplicação responsável pela recolha de dados:

1. A aplicação é auxiliada por um ficheiro de configuração que armazena, em cada linha, o nome de utilizador e o identificador do último *tweet* extraído para aquele utilizador. A aplicação lê o conteúdo deste ficheiro e chama a função "**GET statuses/user_timeline**" iterativamente sobre cada utilizador, que vai preenchendo uma lista de *tweets* juntamente com a sua data de publicação, e identificador.
2. Após terem sido recolhidos todos os *tweets* novos, estes são escritos num ficheiro de texto em que cada linha está no formato <data de publicação, *tweet*, identificador>.
3. Finalmente, para cada utilizador é registado o identificador do *tweet* extraído mais recente e a entrada do ficheiro de configuração para este utilizador é atualizada.

Para garantir que houvesse coerência na relação entre os *tweets* extraídos, definimos três épocas para a recolha de dados, conforme será explicado na secção 4.2.

3.4 Abordagens Específicas

3.4.1 Filtros

Temos como objetivo reduzir a quantidade de dados que o analista de segurança tem de analisar. Assumimos que um *tweet* só pode ser relevante para identificar uma ameaça a uma plataforma se incluir o nome da plataforma (ou alguma alcunha ou acrónimo). Um *tweet* que relate os mais recentes resultados eleitorais não tem interesse para nós, então é descartado.

O filtro inicial de palavras-chave serve para garantir que o analista de segurança classifica apenas *tweets* relacionados minimamente com a infraestrutura de TI que quer proteger. Na ausência deste filtro, o analista teria de percorrer milhares de *tweets*, tarefa que demoraria muito tempo.

Esta abordagem também garante um melhor controlo sobre o equilíbrio de dados positivos e dados negativos. Sem filtro, teríamos conjuntos de dados bastante desequilibrados, pois o número de *tweets* negativos seria substancialmente maior que o número de *tweets* positivos. Reduzimos assim a complexidade do problema, pois só vamos trabalhar sobre os *tweets* relacionados com a infraestrutura que queremos proteger, tanto na fase de classificação manual, quanto na fase de classificação automática.

Na prática, verificamos cada linha do ficheiro de texto produzido na recolha de dados em busca de alguma palavra-chave de cada filtro. Se houver alguma correspondência, escreve-se essa linha num novo ficheiro, que irá conter apenas os *tweets* que mencionem a infraestrutura de TI.

O uso de filtros diferencia o trabalho realizado de outros, pois as palavras-chave são usadas para restringir o âmbito a uma determinada parte da infraestrutura de interesse e não para classificar completamente.

3.4.2 Classificação Manual

Os algoritmos de aprendizagem automática supervisionados usam exemplos de dados rotulados para criar modelos que preveem resultados para dados não rotulados. A classificação manual dos dados tem de ser rigorosa ou o modelo criado não será capaz de produzir os resultados esperados. Para facilitar o processo de classificação manual, criámos uma pequena aplicação com um ambiente gráfico (figura 7) que permite ao analista classificar facilmente os *tweets* de um determinado conjunto de dados. O analista irá observar o *tweet* apresentado e atribuir uma classificação, positiva ou negativa, consoante a relevância do *tweet* ao problema. Quando o analista escolher a classificação atribuída, o próximo *tweet* do conjunto de dados irá ser carregado.

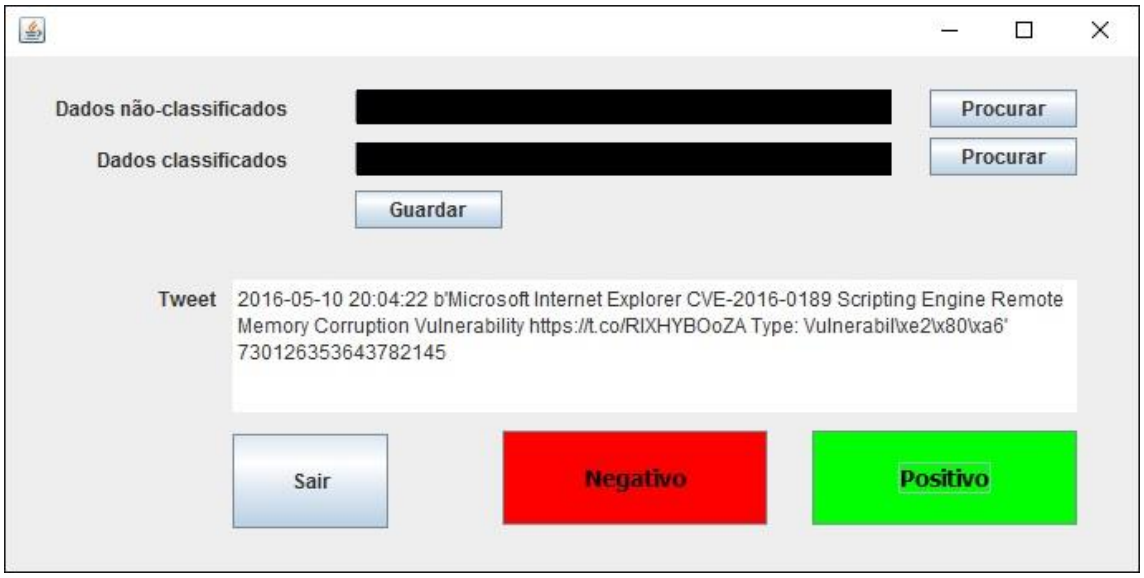

Figura 7: Interface da aplicação que auxilia a classificação manual.

Esta fase de classificação manual pode ser bastante demorada, mas é inevitável para construir máquinas de vetores de suporte. Portanto é essencial reduzir o esforço dedicado pelo analista nesta fase ao aplicar os filtros referidos na secção anterior. Idealmente esta fase deverá continuar até haver uma quantidade de *tweets* classificados que permita produzir classificadores que tenham um bom desempenho para a tarefa em questão. É

também necessário que sejam classificados *tweets* publicados por uma grande variedade de contas diferentes para que o classificador não seja treinado com dados que seguem um formato específico e portanto, que não produza resultados tendenciosos.

Os critérios que levam o analista a classificar os *tweets* como positivos ou negativos vão sempre depender da organização a ser protegida e das necessidades da dita organização. Pode-se atribuir classificação negativa a *tweets* que mencionem uma vulnerabilidade a um determinado produto usado na organização mas que não é relevante para o contexto do problema pois a versão do produto usada está mais atualizada que aquela referida nos *tweets*. Também se pode considerar *tweets* que mencionem o lançamento de um *patch* para um determinado produto como sendo positivos se o objetivo for também receber informação sobre o lançamento de *patches*. Será sempre o trabalho do analista modelar a máquina de vetores de suporte de acordo com o tipo de informação mais adequado às suas necessidades.

3.4.3 Tratamento de Dados

Vimos na secção 3.3 que os dados retirados do Twitter são guardados no formado <data de publicação, *tweet*, identificador>, como consta no seguinte exemplo:

2015-12-02 18:00:08,b'#cybersecurity 4 Government Tech Priorities That Need a Funding Boost Next Year https://t.co/rR57USXHGB #infosec',672113030830788609

Os parâmetros da data de publicação e identificador no contexto do nosso sistema servem apenas para produzir dados estatísticos e organizar este conteúdo. O parâmetro que importa verdadeiramente para a criação do classificador é o *tweet*, mas mesmo assim, este contém alguma informação que é irrelevante e só prejudica a tarefa de aprendizagem da máquina de vetores de suporte.

Para processar um *tweet*, começamos por retirar quase todos os caracteres especiais, hiperligações e as palavras mais frequentes do vocabulário inglês, pois não trazem qualquer significado por si próprias [7]. Não retirámos os pontos e hífens pois podem ser úteis para determinar a versão de um determinado produto (ex. Google Chrome 1.27). Também transformámos todos os números no seu formato por extenso e convertemos todas as letras maiúsculas para minúsculas. Após estas transformações, a partir do exemplo anterior ficamos com o seguinte:

cybersecurity four government tech priorities need funding boost next year infosec

Com os *tweets* neste formato, aplicamos os métodos de extração de características, referidos no capítulo 2 de modo a produzir um vetor que represente o *tweet*. Para os seguintes exemplos, representamos o *tweet* apresentado num vetor de tamanho 200 para o método Word2Vec e um vetor de tamanho 3000 para o método TF-IDF. O segundo

vetor está representado pelo tamanho, índices preenchidos e conteúdo dos índices preenchidos.

- Word2Vec:

[-0.04508161832663146,-0.03790176253427159,-0.011433850381184708, -0.1309715384109454, -0.07126037082211538, 0.0677082110196352, ..., -0.018857966600493953,-0.10372458745471456]

- TF-IDF:

(3000,

[929,990,1211,1291,1445,2094,2302,2402,2893,2905,2907],

[4.942276203131206, 5.086287793647218, 5.737065203168149, 5.872469840174352, 5.54676970053505, 5.5431660930317515, 4.558746171889297, 5.934345243892439, 5.238542019274378, 4.450567051587094, 4.384070256117308])

3.4.4 Classificadores

A infraestrutura informática de uma organização pode ser dividida em muitas plataformas e subsistemas. Desta forma, a abordagem proposta pode ser usada com um único classificador treinado com dados obtidos por filtros contendo a descrição de toda a plataforma ou com vários classificadores específicos para diferentes partes da infraestrutura. Este último caso corresponde a um problema de classificação multi-classe implementado através de vários classificadores binários.

Nesta segunda abordagem cada classificador será treinado apenas com dados relacionados com a sua classe de produtos/sistemas. Isto permite ao analista de segurança alguma flexibilidade na gestão das ameaças, podendo facilmente criar classes com diferentes níveis de prioridade. Por exemplo, numa organização em que as ameaças aos sistemas operativos são muito mais relevantes que as ameaças aos editores de texto, é conveniente que os alertas sejam apresentados ao analista pela sua ordem de importância.

A divisão da infraestrutura em várias classes também pode trazer benefício no desempenho da classificação, ao reduzir a abrangência das classes positivas e negativas. Na figura 8 podemos ver um exemplo de uma organização que tem três produtos (A, B e C) e diferentes métodos de criar os classificadores. O primeiro método é criar um classificador a partir de *tweets* positivos e negativos em relação a qualquer produto da organização. O segundo método passa por criar três classificadores, sendo cada um deles responsável pelo seu produto.

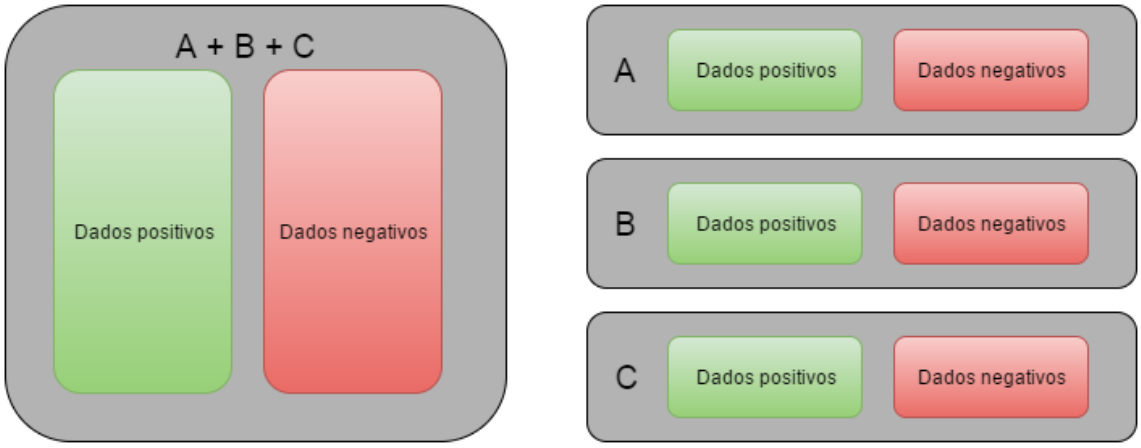

Figura 8: Diferentes métodos de segmentar os dados para criar classificadores.

Como podemos ver, a quantidade de dados para criar os classificadores vai sempre variar consoante a abordagem escolhida e isso pode refletir no desempenho do classificador. Supondo que os dados positivos e negativos foram rotulados com rigor pelo analista, a máquina de vetores de suporte do primeiro método será criada a partir de uma quantidade maior de vetores, que sendo a representação de *tweets* positivos e negativos de todos os produtos da organização, estarão mais distribuídos no espaço vetorial e portanto, o hiperplano ótimo que separa as classes positivas e negativas será mais difícil de determinar. No segundo método, a máquina de vetores de suporte para cada um dos produtos A, B ou C será apenas criada a partir dos dados positivos e negativos de um produto, sendo mais fácil de determinar o hiperplano ótimo pois os vetores de cada classe estão mais agrupados no espaço vetorial.

3.5 Treino dos Modelos

Nesta secção iremos detalhar todos os componentes e interações que estão integrados no modo de treino dos modelos das máquinas de vetores de suporte (MVS). Os classificadores foram criados usando a implementação das máquinas de vetores de suporte lineares da biblioteca MLlib do projeto Apache Spark, na linguagem de programação Java. Como tal, iremos usar RDDs (explicadas na secção 2.5), as estruturas de dados que oferecem paralelização de processamento e tolerância a faltas e explicar o fluxo de dados desde o fim da fase de classificação manual até à criação dos classificadores, também exemplificado na figura 9. Na tabela 1 estão os argumentos que têm de ser fornecidos pelo utilizador para executar a aplicação em modo de treino.

Inicialmente, a aplicação carrega o ficheiro com os *tweets* e os respetivos rótulos para um RDD de tipo String. É efetuada uma transformação que divide este RDD em

dois, um que armazena os *tweets* e outro que armazena os rótulos. Ao RDD contendo os *tweets*, são executados os métodos de tratamento de dados explicados na secção 3.4.3.

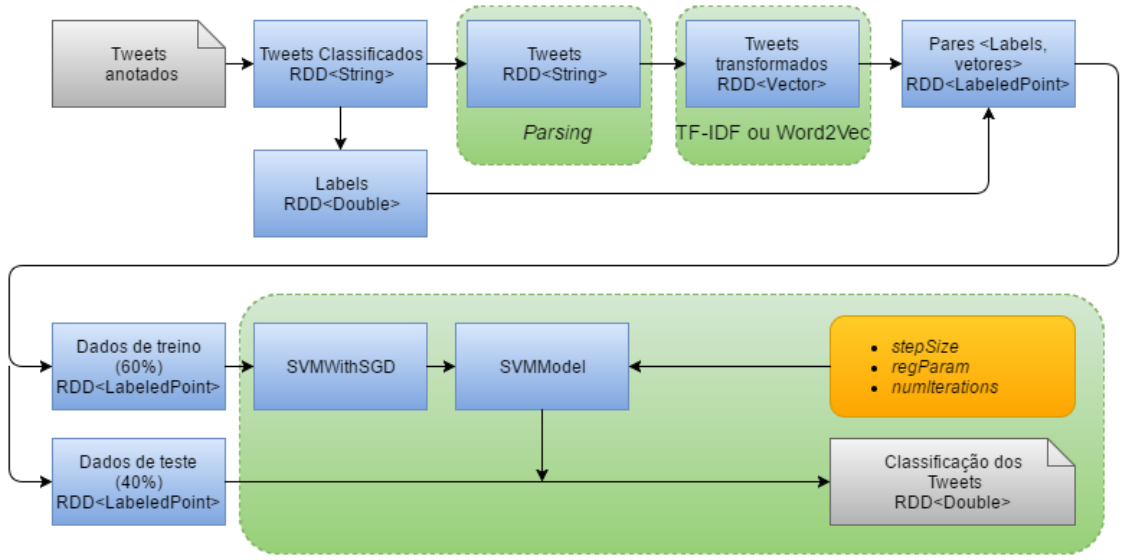

Figura 9: Fluxo de execução da aplicação em modo de treino.

Argumento	**Descrição**
fMode	Modo de extração de características, que pode ter como valor *tfidf* ou *word2vec*.
word2vecPath	Caminho para o modelo Word2Vec. (Apenas quando usado Word2Vec)
numFeatures	Tamanho dos vetores criados pelo TF-IDF. (Apenas quando usado TF-IDF)
labeledData	Caminho para o ficheiro de dados rotulados pelo analista.
stopwordsFile	Caminho para ao ficheiro que contém as palavras que vão ser descartadas na fase de tratamento de dados.
numIterations	Número de iterações da otimização usada na criação do modelo MVS.

Tabela 1: Argumentos da aplicação no modo de treino.

A implementação das máquinas de vetores de suporte usadas requer que os dados de treino sejam *LabeledPoints*, ou seja, um par <*Double*, *Vector*> que representa um vetor e o rótulo atribuído a esse vetor. Ao RDD que contém os *tweets*, é aplicada uma transformação TF-IDF ou Word2Vec, transformando cada *String* no vetor correspondente. O RDD dos vetores e o RDD dos rótulos são transformados num RDD de *LabeledPoints* que é dividido aleatoriamente em duas partes, sendo uma parte composta por 60% dos dados dedicada para treinar o modelo MVS e a outra parte dedicada para validar a eficácia do modelo treinado, composta por 40% dos dados.

A classe *SVMWithSGD* é responsável pela criação dos modelos MVS através do seu método *train()*, que tem os seus argumentos definidos na tabela 2.

Argumento	Descrição
input	RDD<*LabeledPoint*> com os pares <rótulo, vetor> que representam os dados já classificados que vão ser usados para treinar a máquina de vetores de suporte.
stepSize	Valor usado em cada iteração da optimização *Stochastic Gradient Descent*.
regParam	Parâmetro de regularização que ajuda a afinar a classificação da MVS, também conhecido por C.
numIterations	Número de iterações que a optimização *Stochastic Gradient Descent* vai correr.

Tabela 2: Argumentos do método train.

O resultado é um objeto da classe SVMModel, que será usado para classificar novos vetores através do método *predict()*, que contém o seguinte argumento:

- *testData*, vetor que representa um *tweet*.

Esta implementação da MVS usa *Stochastic Gradient Descent* [20], que é um algoritmo de otimização para a sua função de custo. Em cada iteração, em vez de se computar a função de custo com cada vetor do conjunto de dados de treino, é utilizada apenas um subconjunto dos dados de treino. Desta forma, é possível reduzir a complexidade computacional quando o conjunto de dados de treino tem um elevado volume ou quando os dados estão distribuídos. O argumento *stepSize* define a amplitude do passo dado na direção do gradiente em cada iteração. Caso o *stepSize* seja demasiado grande, pode-se falhar o mínimo da função de custo mas se o *stepSize* for demasiado pequeno, o algoritmo irá demorar mais a convergir. Por isso é necessário escolher o

stepSize ideal para criar um modelo que ofereça bons resultados. Outro argumento que requer atenção é o C, que ajuda a melhorar o processo de treino da MVS. Existem casos em que os dados de treino não conseguem ser separados no espaço vetorial [11], então pode se aplicar uma função de custo que ignora vetores que estejam do lado errado da margem de separação. O parâmetro C define o equilíbrio entre uma classificação que segue à regra o modelo criado com os dados de treino (baixo valor de C) e uma classificação que tem melhor desempenho nos dados de teste uma vez que pode ignorar alguns vetores dos dados de treino (alto valor de C). Por fim, o parâmetro *numIterations* tem de ser alto o suficiente para permitir a convergência do modelo, mas não demasiado alto de modo a que afete o desempenho.

Existem então vários parâmetros que podem ser ajustados para melhorar os resultados da classificação conseguidos pelo modelo. A forma de encontrar os melhores parâmetros não é certa e tem de ser feita por via experimental. De modo a encontrar a melhor combinação de valores é necessário fazer várias tentativas alterando o valor do C e do *stepSize*. Definimos ao início um conjunto de valores para testar o *stepSize* e para cada um destes valores, iniciamos a procura pelo valor de C. Começando num valor baixo para C, treinamos o modelo, avaliamos com o conjunto de dados desconhecido e registamos os resultados. De seguida, aumentamos o valor de C e repetimos o processo. Continuamos a fazer isto até chegarmos a resultados satisfatórios que não conseguem ser melhorados. A forma de avaliar cada modelo para cada valor de C é a seguinte:

1. Para cada vetor dos dados de teste, é chamado o método *predict()* que vai devolver um valor entre 1 e -1. De seguida, compara-se cada valor devolvido pelo método *predict()* com o valor do rótulo anexado ao vetor. Se o valor for o mesmo, então o vetor foi classificado corretamente e portanto podemos aumentar a contagem dos verdadeiros positivos ou verdadeiros negativos, dependendo se o vetor foi classificado como positivo ou negativo. Se o valor não for o mesmo, então o vetor foi classificado incorretamente e portanto podemos aumentar a contagens dos falsos positivos ou falsos negativos.
2. Após contagem de todos os resultados, são aplicadas as equações descritas na secção 4.4 para calcular a taxa de verdadeiros positivos (TPR) e a taxa de verdadeiros negativos (TNR) do modelo.
3. Caso este modelo tenha tido os melhores resultados até à fase da sua avaliação, é guardado temporariamente.

Finalmente, o modelo que apresentou os melhores resultados na classificação sobre os dados de teste é guardado em disco, para ser usado mais tarde no modo de operação.

3.6 Operação

Nesta secção iremos explicar o modo de operação da ferramenta (figura 10). Na tabela 3 estão os argumentos que têm de ser fornecidos pelo utilizador para executar a aplicação em modo de operação.

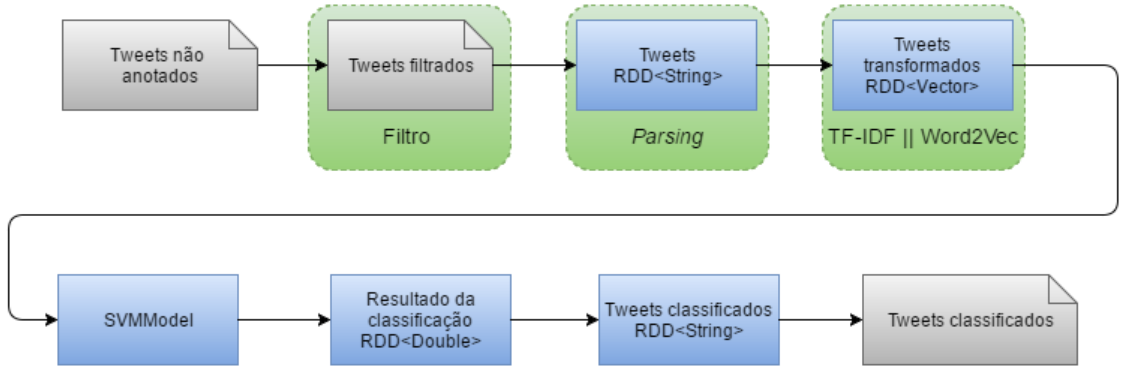

Figura 10: Fluxo de execução da aplicação em modo de operação.

No arranque da aplicação será carregada a máquina de vetores de suporte escolhida no modo de treino. O novo ficheiro de dados irá conter qualquer *tweet* recolhido, independentemente da estratégia definida pelo utilizador. Portanto, antes de os dados serem tratados, serão filtrados de acordo com as palavras-chave definidas pelo utilizador, que deverão representar componentes da sua infraestrutura de TI. De seguida, é carregado o ficheiro que contém os *tweets* sem classificação filtrados para um RDD de tipo *String*. A esse RDD é aplicado o tratamento de dados, composto pelo *parsing* e pelo método de extração de características escolhido, transformando-se num RDD do tipo *Vector*. A máquina de vetores de suporte pode agora classificar os *tweets* em formato vetorial, gerando um RDD<Double> que contém os resultados da classificação de cada *tweet*. Por conveniência, fazemos ainda uma última transformação, adicionando à frente de cada resultado o *tweet* correspondente. Finalmente, escrevemos este RDD em disco para poder ser examinado pelo analista.

3.7 Considerações Finais

Neste capítulo descrevemos o funcionamento da aplicação desenvolvida através da relação dos seus diversos componentes. Percebemos que se pode tornar vantajoso dividir uma infraestrutura de TI em várias partes, sendo que cada parte está associada a um filtro, o que reduz não só o volume de dados que um analista tem de classificar, mas também o esforço computacional de processar uma grande quantidade de dados. Também discutimos os fatores que influenciam o processo de treino das máquinas de vetores de suporte, nomeadamente a necessidade do tratamento de dados para obter uma boa representação vetorial, o rigor por parte do analista em rotular os *tweets* corretamente na

fase de classificação manual. Finalmente incluímos com grande detalhe o funcionamento do modo de treino e do modo de operação, descrevendo com precisão todo o fluxo de dados, os argumentos necessários para a execução e ainda alguns pormenores importantes, como a procura dos parâmetros ideais.

Argumento	**Descrição**
fMode	Modo de extração de características, que pode ter como valor *tfidf* ou *word2vec*.
word2vecPath	Caminho para o modelo Word2Vec. (Apenas quando usado Word2Vec)
numFeatures	Tamanho dos vetores criados pelo TF-IDF. (Apenas quando usado TF-IDF)
newData	Caminho para o ficheiro de dados que vão ser classificados.
stopwordsFile	Caminho para ao ficheiro que contém as palavras que vão ser descartadas na fase de tratamento de dados.
filterwordsFile	Caminho para o ficheiro que contém as palavras-chave do filtro que deve ser aplicado.
svmPath	Caminho para o modelo MVS.
numIterations	Número de iterações da otimização usada na criação do modelo MVS.

Tabela 3: Argumentos da aplicação no modo de operação.

Capítulo 4

Experiências

Neste capítulo apresentamos o trabalho experimental, aplicando a ferramenta desenvolvida num ambiente que simula o seu uso por parte de uma organização que pretende defender a sua infraestrutura de TI. Descrevemos todas as fases necessárias para elaborar a experiência, começando pela construção dos conjuntos de dados usados para o treino e teste dos classificadores, passando pelas métricas usadas para avaliar os classificadores e ainda o papel dos filtros na redução do volume de dados. Incluímos testes com as duas abordagens de extração de características, TF-IDF e Word2Vec, de modo a discutir o desempenho de cada uma. Também efetuamos testes sobre diversos conjuntos de dados, extraídos em momentos diferentes, para avaliar a coerência dos classificadores ao longo do tempo. Por fim incluímos um teste que utiliza filtros, mas não utiliza máquinas de vetores de suporte, para avaliar a necessidade de usar as técnicas que projetámos desde o início do trabalho e confirmar as vantagens da metodologia desenvolvida.

4.1 Contas do Twitter

No contexto do deste projeto, precisávamos de informação relacionada com a área da segurança informática, mais precisamente sobre vulnerabilidades e ameaças. Como forma de adquirir *tweets* que vão ao encontro do nosso objetivo, implementámos inicialmente uma seleção das contas de utilizadores que íamos seguir. Considerando apenas os utilizadores mais predominantes pela sua experiência em segurança informática e pelas suas contribuições para a descoberta e partilha de novas vulnerabilidades e ameaças de segurança. Estes utilizadores são administradores de sistemas, programadores, analistas e consultores de grandes empresas de segurança, investigadores, *freelancers* e "*hackers*". Também incluímos grandes empresas de *software* e *hardware*, instituições divulgadoras de notícias, entidades de resposta a incidentes governamentais e não-governamentais e por fim, contas de certas plataformas que publicam vulnerabilidades, como os arquivos e bases de dados *online* de *exploits*.

O Twitter oferece esta facilidade em encontrar conteúdo relacionado com o nosso projeto devido à relação que há entre os utilizadores. Quando encontrávamos um utilizador que publicava assuntos relevantes, facilmente encontrávamos utilizadores relacionados que publicavam o mesmo tipo de conteúdo. Este processo de escolher as contas de utilizadores é bastante delicado, pois no início do projeto era essencial ter um

conjunto de contas relevantes para ir recolhendo os dados ao longo dos meses. A técnica usada para procurar contas essenciais foi a seguinte:

1. Eram feitas pesquisas no Twitter sobre vulnerabilidades de segurança usando termos como #vulnerability, #0day e #0dayvuln. A figura 11 apresenta exemplos de publicações obtidas nas pesquisas.
2. Registávamos os utilizadores que publicavam esse conteúdo.
3. Verificávamos se as restantes publicações de cada utilizador iam de encontro ao nosso objetivo.
4. Verificávamos as publicações de contas que seguiam estes utilizadores ou que eram seguidas por estes utilizadores.

Figura 11: Exemplo de tweets publicados.

A lista de contas inicialmente obtida é apresentada na tabela 4.

inj3ct0r	TrustedSec	Anomali	briankrebs
Secunia	exploitdb	alienvault	slashdot
dstrom	Info_Sec_Buzz	vuln_lab	threatintel
dangoodin001	ivspiridonov	ThreatFeed	pikisec
SANSInstitute	johullrich	drericcole	F1r3h4nd
MaldicoreAlerts	USCERT_gov	gcluley	hal_pomeran
SecurityWeek	SecurityNewsbot	sans_isc	e_kaspersky

Tabela 4: Primeira lista de contas usadas para a recolha de dados do Twitter.

Mais tarde, devido à necessidade de obter mais *tweets* para efetuar experiências mais elaboradas e avaliar o desempenho do nosso projeto, adicionámos um conjunto ainda maior de contas, contido na lista apresentada na tabela 5.

TenableSecurity	securitywatch	securityaffairs	zer0element
notsosecure	CyberExaminer	SCMagazine	DMBisson
lennyzeltser	IT_securitynews	teamcymru	WordPress
MicrosoftEdge	JoomlaTips	sjzaib	SecurityMagnate
Cisco	Dell	linuxtoday	securityninja
cyberopsy	OWASP_Java	_WPScan_	d_plusk
threatpost	Rootsector	Microsoft	linuxfoundation
ChidoDike	Sec_Cyber	ptracesecurity	msftsecurity
LinuxSec	hack3rsca	CiscoSecurity	NytroRST
joomla	Windows	crackerhacker00	fstenv
HPE_Security	googlechrome	wordpressdotcom	packet_storm
RokaSecurity	Oracle	firefox	wpbeginner
YoKoAcc	SecurityCrap	jasonlam_sec	threatmeter

Tabela 5: Segunda lista de contas usadas para a recolha de dados do Twitter.

Na próxima secção iremos descrever os conjuntos de dados usados para avaliar a metodologia desenvolvida, que foram obtidos através das contas referidas nesta secção.

4.2 Conjuntos de Dados usados

Foram usados três conjuntos de dados distintos no trabalho experimental. O primeiro conjunto de dados (**d1**) foi usado para produzir e validar os classificadores. Os dois restantes conjuntos de dados (**d2** e **d3**) foram usados para avaliar a solução proposta. O conjunto **d1** contempla *tweets* recolhidos a partir das 28 contas apresentadas na tabela 4, extraídos entre 1 de Novembro de 2015 e 1 de Abril de 2016. O conjunto **d2** engloba *tweets* recolhidos a partir das 28 contas da tabela 4, extraídos entre 1 de Abril de 2016 e 15 de Maio de 2016, e das restantes contas na tabela 5, extraídos entre 1 de Novembro de 2015 e 15 de Maio de 2016. O conjunto **d3** foi elaborado a partir das mesmas contas usadas no conjunto **d2**, mas os dados foram recolhidos entre 15 de Maio de 2016 e 10 de Julho de 2016. O conjunto de dados **d1** contém 71024 *tweets* uma vez que o processo de

recolha foi mais frequente, o que permitiu ultrapassar as restrições da API do Twitter. Os conjuntos **d2** e **d3** contêm, respetivamente, 57579 *tweets* e 66608 *tweets* porque estes foram extraídos ao longo de um período de tempo menor, no qual as restrições da API do Twitter impediram a recolha dos *tweets* mais antigos de cada conta.

4.3 Configurações

Temos como objetivo, através de experiências efetuadas, mostrar a viabilidade da nossa aplicação. Como tal, criámos um ambiente capaz de simular o seu uso num cenário real.

Foram construídos 5 classificadores diferentes, tendo cada o propósito de classificar *tweets* relevantes para uma parte de uma infraestrutura de TI numa organização ficcional. A razão que levou à escolha destas plataformas foi a quantidade de dados disponível para produzir os classificadores. Com uma pequena amostra de dados, seria muito difícil obter boa convergência no treino e boa capacidade de generalização dos classificadores, portanto, para efeitos de avaliação, escolhemos as infraestruturas de TI de modo a lidar com o problema da escassez de dados. Foram então considerados diferentes classificadores que têm o objetivo de descobrir ameaças de segurança às seguintes plataformas:

- A: Produtos da Oracle e Cisco, usando um filtro com as palavras-chave ‘oracle’ e ‘cisco’;
- B: Browsers mais usados (Firefox, Chrome, IE, Edge), usando um filtro com as palavras-chave ‘google chrome’, ‘internet explorer’, ‘firefox’, ‘microsoft edge’ e ‘edge’;
- C: Gestores de conteúdo mais usados (Wordpress e Joomla), usando um filtro com as palavras-chave ‘wordpress’, ‘joomla’ e ‘wp’;
- D: Sistemas Operativos (Windows e Linux), usando um filtro com as palavras-chave ‘microsoft windows’, ‘ms’, ‘linux’ e ‘operating system’;
- A+B+C+D: União de todas as plataformas anteriores.

Com esta escolha de plataformas, conseguimos de forma realista representar uma infraestrutura informática.

4.4 Métricas

As métricas para este caso de estudo foram a TPR (*True Positive Rate*), TNR (*True Negative Rate*) e a Precisão. A TPR representa a percentagem de *tweets* que são corretamente classificados como positivos. Uma elevada TPR significa que o classificador tem um elevado grau de sucesso a identificar os *tweets* que representam ameaças para a infraestrutura de TI. No entanto uma boa TPR apenas não é suficiente,

pois um classificador que defina qualquer *tweet* como relevante irá atingir uma TPR de 100% mas também irá apresentar um excesso de informação irrelevante ao analista. Assim, temos de garantir que a TNR seja também elevada, ou seja, o classificador tem de classificar corretamente os dados negativos para que estes não sejam enviados para o analista. Finalmente a Precisão representa a percentagem de *tweets* realmente positivos que estão dentro do conjunto de *tweets* que o classificador previu como positivos. Esta métrica mede a quantidade de dados erradamente classificados como positivos. As equações das métricas usadas são as seguintes:

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

$$Precisão = \frac{TP}{TP + FP}$$

Sendo:

- TP (*True Positives*): Verdadeiros positivos, representam os *tweets* **relevantes** que foram classificados como positivos.
- TN (*True Negatives*): Verdadeiros negativos, representam os *tweets* **não-relevantes** que foram classificados como negativos.
- FP (*False Positives*): Falsos positivos, representam os *tweets* **não-relevantes** que foram classificados como positivos.
- FN (*False Negatives*): Falsos negativos, representam os *tweets* **relevantes** que foram classificados como negativos.

4.5 Representação dos Dados

Como vimos na secção 2.3, a transformação dos dados em vetores é um passo fundamental para qualquer tarefa que envolva aprendizagem automática. Esta representação vetorial terá de ser boa para ser possível criar as MVS que vão classificar os dados para a avaliação.

Usando o Word2Vec, um dos requisitos é a existência de um conjunto vasto de dados que será usado para criar o modelo de referência e permitirá transformar novas cadeias de texto em vetores. O conjunto de dados usado, que afetará a qualidade da representação dos dados é uma parte de um ficheiro com os primeiros mil milhões de caracteres da

Wikipédia [8], composto por apenas os primeiros 300 milhões de caracteres. Também definimos o tamanho dos vetores transformados em 200. A escolha da dimensão dos vetores é importante para determinar o desempenho do modelo, sendo que uma maior dimensão irá melhorar a representação dos dados [9]. Estas duas decisões devem-se à quantidade de recursos computacionais necessários para poder processar a criação do modelo, que no posto de trabalho usado para desenvolver este trabalho eram insuficientes.

Para criar o modelo TF-IDF usávamos sempre o próprio conjunto de dados (**d1**, **d2**, ou **d3**). A diferença entre vetores transformados por um modelo gerado a partir de diferentes conjuntos de dados está apenas na pontuação atribuída a cada elemento do vetor. Duas frases iguais mas transformadas por modelos diferentes irão ter os mesmos elementos mas com valores diferentes.

4.6 Treino dos Classificadores

O processo de treino dos classificadores requer um conjunto de *tweets* corretamente classificados de acordo com as plataformas consideradas. Dos 71024 *tweets* do conjunto **d1**, aplicámos um filtro para cada arquitetura de modo a reduzir o volume de dados a classificar. Para o classificador A foram filtrados 1070 *tweets*, para o classificador B 714 *tweets*, para o classificador C 1092 *tweets*, e para o classificador D 1132 *tweets*. Consequentemente, reduziu-se a quantidade de *tweets* a serem manualmente classificados como positivos ou negativos, de 71024 para 4008.

Com os *tweets* em número reduzido, prosseguimos com a classificação manual. Para esta experiência, considerámos como positivos os *tweets* que realmente apresentam ameaça aos produtos de cada arquitetura, ou seja, aqueles que mencionem algum ataque ou alguma vulnerabilidade que esses produtos tenham. Como estamos mais interessados em problemas e não em soluções, decidimos evitar a classificação positiva dos *tweets* que mencionem atualizações aos produtos (como podemos ver na última linha da tabela 6) pois assumimos que o analista terá sempre acesso a esse tipo de informação a partir de outras fontes. Na tabela 6, estão exemplos de diferentes *tweets* que foram classificados como positivos ou negativos. Conseguindo produzir um conjunto de dados relevante e corretamente classificado, podemos prosseguir para o treino dos classificadores.

Como referido anteriormente, quisemos avaliar o desempenho dos classificadores quando usados métodos de extração de características diferentes, para determinar a importância da representação dos *tweets*. Então para a execução da nossa aplicação sobre cada conjunto de dados **d2** e **d3** criámos dois classificadores, um considerando a técnica Word2Vec, o outro considerando a metodologia TF-IDF.

Classificação	***Tweet***
1	Cisco Products Remote Command Execution Vulnerabilities https://t.co/1Ea656H5rD CGI script in Cisco FX-OS before 1.1.2 on Firepower 9000
1	Windows Built-In PDF Reader Exposes #Edge Browser to Hacking | Softpedia #infosec https://t.co/5zXJilpfXE
1	[dos] - #Linux (Ubuntu 14.04.3) - perf_event_open() Can Race with execve() (/etc/shadow) https://t.co/oYgEW9e8wg #ExploitDB
-1	NEWS: #Oracle Delivers 10x Performance Improvement in #Cloud Network Infrastructure: https://t.co/uJviiKCJnG #OracleCloud
-1	Did you get a new Windows computer over the holidays? Here's how to set Chrome as your default browser #chrometip https://t.co/9zacezmGXB
-1	Patch now! #Google fixes multiple high-severity #vulnerabilities in #Chrome 49 for #Windows, #Mac, #Linux

A fase de criação dos classificadores é complexa e necessita da escolha adequada de parâmetros, tal como vimos na secção 3.5. Os valores escolhidos para os parâmetros *stepSize* e *numIterations* foi fixado para a criação de todos os modelos apresentados neste capítulo. Decidimos colocar estes parâmetros no seu valor padrão pois a quantidade de dados que temos não justifica a sua alteração, sendo que o desempenho (a nível de tempo de processamento) também não varia significativamente com a diminuição dos valores. O valor padrão do *stepSize* é de 1 e o valor padrão do *numIterations* é de 100.

O valor do parâmetro C (*regParam*) varia consoante o objetivo de melhorar os resultados do modelo nos dados de teste. Um pormenor importante é saber quais as condições impostas para descobrir o melhor modelo. Como explicado na secção 3.5, é efetuada uma procura do melhor valor de C que consiga produzir um modelo adequado. O modo de procura implementado varia consoante o método de extração de características usado, mas baseia-se sempre em incrementos sucessivos no valor de C.

Caso seja usado TF-IDF, os incrementos sucessivos de C são constantes, começando em 0 e progredindo até 2, com incrementos sucessivos de 0.1 em cada teste. No caso do Word2Vec, desenhámos um pequeno algoritmo de procura que funciona da seguinte maneira:

Tabela 6: Exemplos de tweets classificados como positivos (1) ou negativos (-1).

1. Define-se como 1 o valor inicial de C e como 1 o valor do incremento.
2. Incrementa-se C sucessivamente registando os resultados obtidos sobre os dados de teste.
3. Quando o valor de C causa resultados nos dados de teste muito diferentes dos da iteração anterior, significa que o valor de C está a subir demasiado e portanto diminui-se o valor de incremento, multiplicando-o por 0.1.
4. Decrementa-se o valor de C para que a alteração dos resultados nos dados de teste seja capturada pelo novo valor de incremento.
5. A procura continua até chegar a um limite de 5 diminuições do valor de incremento, ou seja, quando este chega a 0.00001.

Esta diferença na procura deve-se à representação dos *tweets* no espaço vetorial, que no Word2Vec consiste na atribuição de valores para cada elemento do vetor. Usando TF-IDF, os elementos do vetor só serão preenchidos consoante o número de palavras que o *tweet* tem. No final, para cada *tweet* representado pelo Word2Vec, teremos sempre um vetor de tamanho 200 com todos os seus elementos preenchidos. Estes vetores, quando usados para produzir as MVS, irão dificultar a construção do hiperplano ótimo, sendo neste caso mais importante a otimização do valor de C para obter resultados melhores. Nos *tweets* representados pelo TF-IDF, embora a dimensão dos vetores seja maior, a maioria dos seus elementos terá valor nulo, facilitando assim a construção do hiperplano ótimo, embora ainda possa ser benéfica a otimização do valor de C.

4.6.1 Resultados da Fase de Validação

Para validar as máquinas de vetores de suporte para cada configuração, separaram-se os *tweets* classificados em dois subconjuntos de dados gerados aleatoriamente: um para treino com 60% dos dados, outro para validação após o treino, com os restantes 40%. Para cada *tweet* dos dados de validação o classificador faz uma previsão (positivo ou negativo) que será rotulada como verdadeiro positivo, verdadeiro negativo, falso positivo, ou falso negativo. Estes resultados permitem o cálculo posterior da TPR e da TNR, usadas para avaliar e validar o classificador.

Uma vez que a partição dos dados em conjuntos de treino e de validação é feita aleatoriamente, existe a possibilidade de ser feita uma escolha que não favorece a obtenção de bons classificadores, ou, no caso contrário, de que se faça uma escolha pouco provável que leva a excelentes classificadores. Para garantir que a abordagem é bem avaliada e para que se possam comparar fielmente diferentes estratégias, repetiu-se o processo de treino e validação 10 vezes, gerando em cada uma novos conjuntos de treino e validação. Se os resultados estatísticos forem consistentes, ou seja, se a variação das

métricas não for significativa, então podemos confiar no conjunto de dados classificados e consequentemente no classificador.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
TPR média	95%	90%	94%	90%	92%
TNR média	96%	96%	95%	96%	96%
Precisão média	95%	90%	93%	90%	93%
D. Padrão (TPR)	1.3%	3.1%	2.1%	1.2%	1.7%
D. Padrão (TNR)	1.7%	1.4%	2.0%	2.0%	1.8%
D. Padrão (Precisão)	1.7%	4.1%	2.3%	4.3%	2.4%

Tabela 7: Resultados do processo de avaliação sobre os dados de validação, usando o método de extração de caracteristicas TF-IDF.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
TPR média	100%	100%	100%	99%	97%
TNR média	100%	100%	100%	100%	99%
Precisão média	100%	100%	99%	99%	98%
D. Padrão (TPR)	0.4%	0.4%	0.6%	0.5%	0.5%
D. Padrão (TNR)	0.3%	0.3%	0.3%	0.6%	0.4%
D. Padrão (Precisão)	0%	0%	0.4%	0.9%	0.4%

Tabela 8: Resultados do processo de avaliação sobre os dados de treino, usando o método de extração de características TF-IDF.

O resultado do processo de validação dos classificadores (usando TF-IDF) para cada arquitetura é apresentado na tabela 7. Como podemos observar, cada classificador atingiu na fase de validação valores elevados para a TPR e TNR, com médias entre os 90% e os 95% e baixa variabilidade, permitindo-nos confiar nos classificadores desenvolvidos.

Os resultados da classificação nos dados usados para treinar o classificador são apresentados na tabela 8. Os resultados são praticamente perfeitos, mostrando que o classificador aprendeu a separar praticamente todos os dados rotulados desse conjunto. É de salientar que no caso do classificador da arquitetura completa A+B+C+D, os resultados no conjunto de treino revelam alguma dificuldade em classificar corretamente a totalidade dos dados, o que demonstra o mérito da separação da arquitetura global em arquiteturas parciais mais pequenas e mais simples de abordar por parte dos algoritmos de classificação. Os resultados mostram que o classificador é capaz de generalizar os seus

resultados, uma vez que a partir de um conjunto de dados bem rotulados conseguiu classificar corretamente dados que ainda não tinham sido vistos pelo classificador.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
TPR média	92%	84%	84%	85%	79%
TNR média	80%	81%	73%	84%	78%
Precisão média	50%	68%	74%	53%	55%
D. Padrão (TPR)	1.7%	9.4%	7.8%	2.7%	5.7%
D. Padrão (TNR)	2.4%	5.6%	8.3%	3.1%	6.2%
D. Padrão (Precisão)	3.7%	7.2%	3.6%	4.4%	5.3%

Tabela 9: Resultados do processo de avaliação sobre os dados de validação, usando o método de extração de características Word2Vec.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
TPR média	93%	86%	91%	90%	79%
TNR média	79%	82%	79%	75%	80%
Precisão média	52%	68%	64%	52%	57%
D. Padrão (TPR)	3.4%	1.6%	2.8%	5.9%	3.4%
D. Padrão (TNR)	3.2%	1.7%	2.6%	6.5%	3.2%
D. Padrão (Precisão)	2.9%	2.2%	2.9%	5.4%	5.7%

Tabela 10: Resultados do processo de avaliação sobre os dados de treino, usando o método de extração de características Word2Vec.

No caso do Word2Vec os resultados são apresentados na tabela 9. Na validação com os dados de teste, em relação à abordagem TF-IDF, os classificadores produzidos usando Word2Vec obtiveram resultados inferiores tanto na TPR média como na TNR média, em qualquer das configurações. Para além disso, podemos ver que o desvio padrão atinge valores altos em alguns casos, significando que houve grande variação nos resultados obtidos nas 10 repetições do processo de validação. Embora tenha obtido resultados menos satisfatórios, iremos mesmo assim avaliar os classificadores nos conjuntos de dados **d2** e **d3**. A tabela 10 apresenta os resultados dos classificadores Word2Vec sobre os dados de treino. O comportamento é semelhante ao dos classificadores TF-IDF, no sentido em que os resultados em dados de treino superaram os resultados em dados teste.

Em virtude dos resultados obtidos, nas experiências descritas de seguida os classificadores serão usados com os valores de C apresentados na tabela 11.

Configuração	A	B	C	D	A+B+C+D
C (TF-IDF)	0.3	0.1	0.3	0.1	0.5
C (Word2Vec)	13.27436	13.2737	13.27461	13.27377	13.27438

Tabela 11: Valores de C escolhidos para os classificadores de cada configuração.

4.7 Avaliação

Os conjuntos de dados **d2** e **d3** foram preparados com o objetivo de simular um cenário em que os *tweets* estão a ser recolhidos em tempo real, à medida que são publicados no Twitter. O processo de classificação começa ao aplicar o filtro específico da configuração a cada novo conjunto de dados. Este filtro reduz consideravelmente a quantidade de *tweets* que vão ser classificados. Dos 57579 *tweets* do conjunto de dados **d2** foram filtrados 426 para a configuração A, 532 para a configuração B, 1038 para a configuração C e 2835 para a configuração D. Para a configuração A+B+C+D foram filtrados 4773. Do conjunto de dados **d3**, composto por 66608 *tweets*, foram filtrados 758 *tweets* para a configuração A, 782 para a configuração B, 728 para a configuração e 1568 para a configuração D. Finalmente, a configuração A+B+C+D é composta por 3798 *tweets*. É de notar que a quantidade de *tweets* filtrados na configuração A+B+C+D, para cada conjunto de dados, não é exatamente a soma do número de *tweets* de cada. Isto porque podem haver *tweets* que incluem palavras de dois ou mais filtros diferentes, e nesse caso os *tweets* são incluídos apenas uma vez.

Sobre os *tweets* filtrados foi aplicado o classificador da plataforma correspondente, que irá fazer a previsão para cada *tweet*. No final, temos um *stream* de *tweets* classificados. Para avaliar este processo, verificámos manualmente cada *tweet* rotulado pelos classificadores e através da nossa análise, de acordo com os objetivos de segurança da nossa organização ficcional especificados na secção anterior, atribuímos a cada *tweet* a classificação de verdadeiro positivo, verdadeiro negativo, falso positivo ou falso negativo.

4.7.1 Avaliação usando d2

A figura 12 apresenta os resultados obtidos sobre o conjunto de dados **d2**. À esquerda estão os resultados usando TF-IDF e à direita os resultados usando Word2Vec. São apresentados os valores de TPR, TNR e Precisão. Para fins de completude, a tabela 12 apresenta os verdadeiros positivos e verdadeiros negativos (classificados manualmente) para o conjunto de dados **d2**.

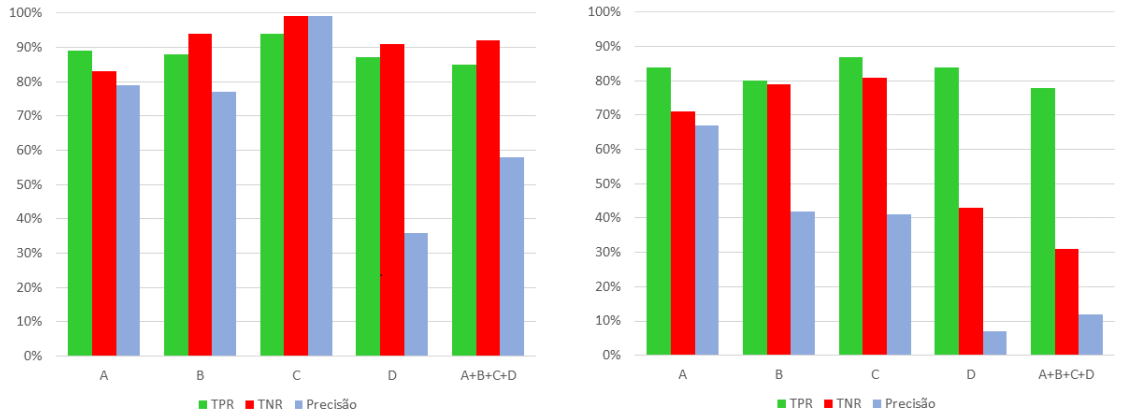

Figura 12: Resultados sobre o conjunto de dados d2. À esquerda estão os resultados usando TF-IDF e à direita estão os resultados usando Word2Vec.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
V. positivos	177	86	138	138	531
V. negativos	249	446	900	2697	4242

Tabela 12: Lista de verdadeiros positivos e verdadeiros negativos (classificados manualmente) para cada configuração do conjunto de dados d2.

Como podemos verificar, os resultados obtidos usando a representação de dados TF-IDF estão próximo dos que foram alcançados na validação dos classificadores. O menor desempenho dos classificadores face aos resultados no processo de treino deve-se provavelmente ao facto da maior parte dos *tweets* que preenchem o conjunto de dados **d2** serem provenientes de contas de utilizador que não foram utilizadas no processo de treino. Esta poderá ser uma fraqueza desta abordagem, que usa classificadores influenciados pelos hábitos dos utilizadores que publicam os *tweets*. Mesmo assim, consideramos os resultados positivos, já que no caso da TPR o pior valor obtido é de 85% para a configuração A+B+C+D. Se forem considerados os quatro classificadores individuais, obtém-se um resultado mais favorável, com uma TPR média próximo de 90%, o que valida a abordagem de dividir a infraestrutura informática em múltiplas infraestruturas mais simples. No caso da TNR, os valores encontram-se acima dos 90% em todas as configurações exceto a A, fazendo-se notar o impacto da diferença dos dados usados no processo de treino e no processo de avaliação. Isto significa que grande parte da informação irrelevante foi bem classificada, o que é uma caraterística desejada numa situação real. Através da Precisão, podemos ver que os classificadores das configurações A,B e C conseguem com sucesso mostrar um baixo número de falsos positivos. O mesmo não acontece com as configurações D e A+B+C+D o que está diretamente relacionado

com o desequilíbrio entre os *tweets* positivos e negativos do conjunto de dados **d2**, pois embora a percentagem de *tweets* mal classificados seja baixa (aproximadamente 8%), esta baixa percentagem representa uma grande quantidade de *tweets* (325).

Quando usada a representação de dados Word2Vec, podemos observar que as TPR são bastante altas, com uma média ligeiramente superior aos 80%. Isto significa que embora os resultados tenham sido inferiores à vertente TF-IDF, continuam a ser bastante positivos na classificação correta de ameaças de segurança. Contudo, a viabilidade dos classificadores fica comprometida quando analisamos a TNR e a Precisão. As TNR de 43% da configuração D e de 31% da configuração A+B+C+D são completamente indesejáveis, pois embora os classificadores dessas configurações tenham conseguido detetar uma alta taxa de verdadeiros positivos, também classificaram incorretamente grande parte dos dados negativos, resultando numa maior quantidade de dados que o analista terá de validar manualmente. Mas através da precisão, podemos também criticar os resultados obtidos na configuração B e C. Em ambos os casos, a Precisão ronda os 40%, quer isto dizer que apenas 42% e 41% dos dados classificados como positivos são realmente ameaças de segurança. Dependendo da qualidade de informação publicada pelas contas do Twitter escolhidas, podem existir situações em que o número de dados negativos é muito maior que o número de dados positivos. Por exemplo, no caso da configuração C, foram filtrados 138 *tweets* que nós considerámos positivos e 900 *tweets* que nós considerámos negativos. Desses 138 *tweets*, 120 (87%) foram classificados como positivos e dos 900 *tweets* negativos 728 (81%) foram classificados como negativos. Mas é também verdade que 172 dos *tweets* negativos foram classificados como positivos. Concluindo o exemplo, mais de metade dos *tweets* que foram classificados como positivos na experiência são, de facto, irrelevantes. Isso motivou-nos a registar também a Precisão, pois como podemos ver, os resultados avaliados apenas com a TPR e TNR seriam insuficientes, induzindo-nos em erro ao considerar a viabilidade dos classificadores das configurações B e C. Estas experiências, usando diferentes representação de dados, favorecem claramente a técnica de extração de características TF-IDF.

4.7.2 Avaliação usando d3

A segunda experiência que fizemos foi aplicar novamente os classificadores sobre o conjunto de dados **d3**, que foi recolhido durante um período de tempo posterior ao da recolha dos dados de **d2**. Os resultados são apresentados na figura 13. A tabela 13 apresenta a lista de verdadeiros positivos e verdadeiros negativos dos *tweets* (classificados manualmente) de cada configuração para **d3**.

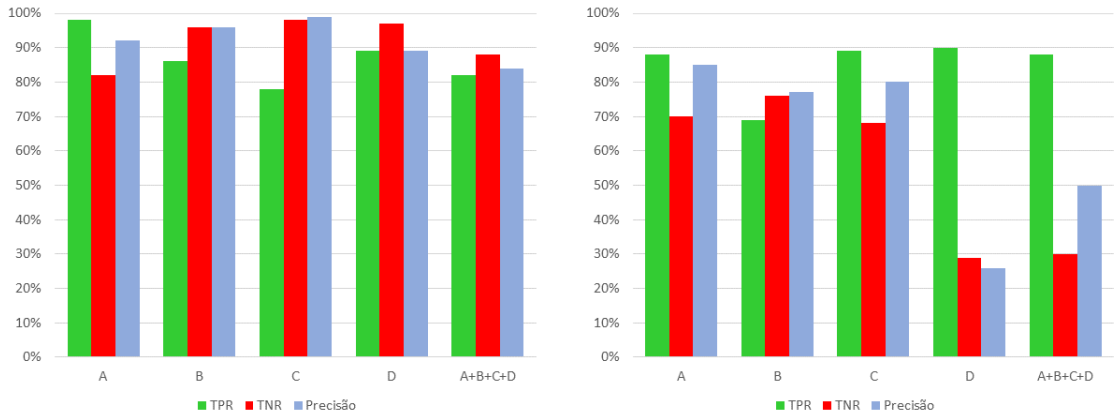

Figura 13: Resultados sobre o conjunto de dados d3. À esquerda estão os resultados usando TF-IDF e à direita estão os resultados usando Word2Vec.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
V. positivos	502	420	425	336	1669
V. negativos	256	362	303	1232	2129

Tabela 13: Lista de verdadeiros positivos e verdadeiros negativos (classificados manualmente) para cada configuração do conjunto de dados d3.

Podemos observar que em ambas as representações de dados os classificadores foram menos eficazes a prever os resultados no conjunto **d3** comparativamente ao conjunto **d2**. Usando TF-IDF, a TPR média das 5 configurações ronda os 86%, valor que ainda consideramos bom tendo em conta o nosso objetivo. A TNR média e a Precisão média entre as configurações estiveram acima dos 90%, demonstrando a consistência dos resultados nestas métricas face à experiência sobre o conjunto de dados **d2**.

Usando Word2Vec, a TPR melhorou em 4 das 5 configurações contudo, a TNR diminuiu em todas elas. Podemos também observar que a Precisão aumentou drasticamente nas configurações B e C face aos resultados em **d2**. Embora haja algumas diferenças entre as experiências, um padrão que se mantém são os péssimos resultados na TNR e na Precisão nas configurações D e A+B+C+D, que se pode justificar pela desadequada representação dos dados usados para treinar e avaliar estes classificadores e que portanto não foram capazes de generalizar os seus resultados.

4.7.3 Discussão

Através destas duas experiências quisemos avaliar o desempenho da nossa solução em três pontos:

- Resultados usando a abordagem de divisão da infraestrutura de TI;
- Comparação dos métodos de extração de características;
- Relação dos resultados obtidos em intervalos de tempo diferentes.

Os resultados são positivos quando usado o método TF-IDF, pois demostram que a solução tem uma boa capacidade de deteção de ameaças sendo capaz de reduzir os dados irrelevantes que são apresentados ao analista. Também foi possível verificar que os resultados se mantêm consistentes entre as experiências, o que mostra que a atualização dos classificadores, embora necessária, não terá de ser feita com uma frequência muito alta.

Em relação ao método Word2Vec, os resultados não superaram as expetativas. Para além de haver uma enorme discrepância de resultados entre configurações, também não foi obtida a consistência desejada entre experiências. Os resultados obtidos devem-se à insuficiência ou qualidade dos dados para treinar os modelos.

A última experiência tenta responder a seguinte questão: *Será que uma abordagem baseada em aprendizagem automática é necessária neste problema?* Assim, fizemos uma experiência sobre o conjunto de dados **d2** usando uma abordagem básica de filtros de palavras ao invés de recorrer às MVS. Desta forma construímos um classificador que substitui essa camada por outro filtro de palavras, composto por palavras-chave relacionadas com segurança informática. Estas palavras são listadas na tabela 14. Considerou-se que um *tweet* seria relevante se para além de incluir uma palavra-chave do filtro da sua configuração (descritos na secção 4.3), também incluísse uma palavra-chave do filtro geral. Os resultados são apresentados na tabela 15 e mostram a percentagem de *tweets* relevantes entre os *tweets* que foram rotulados como positivos.

Estes resultados demonstram que as abordagens muito básicas, como a pesquisa por palavras-chave, estão muito aquém dos algoritmos de aprendizagem automática em tarefas de classificação e relevância no contexto da deteção de ameaças à segurança informática.

xss	dos	hacking	rootkit
csrf	port sweep	threat	security
cross-site request forgery	forgery	malware	botnet
cross site request forgery	sql injection	vulnerability	hack
rxss	command injection	misconfiguration	hacker
sxss	injection	vuln	brute force
stored xss	overflow	exposure	sudo
reflected xss	portscan	cyber attack	cracker
denial of service	cross site scripting	cyber command	trojan
information disclosure	cross site	cyber security	virus
man in the middle	shellcode	cyber terror	navigation
man-in-the-middle	0 day	cybersecurity	webapps
man at the end	0day	redirect	phishing
mat-at-the-end	exploit	administrator	phreaking
certificate	breach	root	zombie

Tabela 14: Palavras-chave do filtro geral.

Configuração	**A**	**B**	**C**	**D**	**A+B+C+D**
Precisão	48%	16%	69%	15%	31%

Tabela 15: Resultado do classificador baseado unicamente em filtros.

4.8 Considerações Finais

Neste capítulo elaborámos três experiências com o objetivo de comprovar a eficiência da solução proposta. Começámos por especificar as contas que serão alvo do processo de recolha de dados. Esta escolha de contas terá de ser bem feita ou então caímos na possibilidade de uma grande fatia dos dados extraídos não ser relevante, o que pode levar a uma maior sobrecarga de trabalho para o analista. De seguida, descrevemos os conjuntos de dados que iríamos usar nas experiências, sendo o primeiro usado para criar os classificadores e os restantes usados para os avaliar. Tivemos como objetivo garantir que as experiências fossem fiéis à realidade, por isso elaboramos uma organização ficcional com um conjunto de produtos que precisavam de ser alvo de monitorização.

Definimos também as métricas que seriam usadas para avaliar os resultados das experiências.

Com todo o ambiente especificado, descrevemos o processo de treino dos classificadores que consistiu em explicar como foram aplicado os filtros das plataformas que queríamos monitorizar, os critérios de seleção dos *tweets* considerados como relevantes, a procura dos parâmetros das MVS e os resultados da validação dos classificadores. Finalmente, efetuamos as experiências sobre os dois conjuntos de dados, o que nos permitiu discutir e tirar algumas conclusões sobre a viabilidade da abordagem. Também incluímos uma experiência considerando uma abordagem de deteção de ameaças menos complexa para demonstrar a necessidade de usar as técnicas que foram aplicadas neste trabalho.

Capítulo 5

Conclusões e Trabalho Futuro

Neste trabalho criamos um filtro baseado em técnicas de aprendizagem automática capaz de detetar publicações no Twitter que identifiquem ameaças de segurança ou vulnerabilidades a sistemas de informação. A nossa arquitetura é composta pela combinação de um módulo de recolha de *tweets* que extraí os dados publicados por utilizadores pré-definidos, um filtro inicial de palavras-chave que contém termos relacionados com os sistemas a defender, o tratamento de dados necessário para representar os *tweets* em formato numérico e o uso de máquinas de vetores de suporte para criar um classificador capaz de distinguir publicações consoante o nível de ameaça que representam para os sistemas informáticos.

Introduzimos os conceitos teóricos dos componentes da nossa proposta e justificamos a sua criação com base na falta de estudos académicos relacionados nesta área e na sua grande importância em adicionar funcionalidades a sistemas de segurança complexos já existentes. Para validar a nossa proposta, recorremos a um conjunto de experiências que coloca a aplicação desenvolvida num ambiente próximo de um cenário real em que existe a necessidade de proteger sistemas de informação. Os resultados destas experiências são positivos, pelo que consideramos que os objetivos foram cumpridos.

Foi possível através do trabalho realizado responder a alguns desafios importantes. A união de um filtro de palavras com a máquina de vetores de suporte permitiu a criação de um classificador eficaz que exige um reduzido esforço humano para o produzir. Usando aprendizagem automática é possível ultrapassar as soluções que se baseiam unicamente em filtros de palavras, devido à capacidade de distinguir *tweets* que embora tenham muitas palavras relevantes, não mencionem qualquer ameaça de segurança relevante. Também foi verificado que a divisão do problema em vários classificadores pode trazer uma vantagem, tanto a nível da organização e apresentação de resultados como também no desempenho dos classificadores, pois serão mais específicos.

Finalmente concluímos que numa era onde os sistemas de informação são cada vez mais utilizados, métodos de inferir resultados com mínima (ou nenhuma) intervenção humana podem ter um grande impacto na eficiência deteção de ameaças de segurança.

5.1 Trabalho Futuro

Este trabalho abre a possibilidade para muitas melhorias futuras. Uma bastante importante seria integrar um mecanismo de *feedback* em que o analista verifica o resultado do classificador e usa os dados corrigidos para criar classificadores melhores. Ao introduzir novos exemplos de *tweets* positivos e negativos para atualizar as MVS, será possível manter a capacidade de classificar com precisão ao longo do tempo.

Outro problema surge quando uma organização que implemente esta solução adicione novos componentes à sua infraestrutura de TI. Para cada alteração na infraestrutura, será necessário criar novos classificadores ou remover classificadores que se tornam inúteis. No caso de usar um classificador para cada partição da infraestrutura, esta tarefa pode ser menos cara, pois basta adicionar um classificador aos existentes. Mas se for usado apenas um classificador para toda a infraestrutura, será necessário atualizar este classificador para que fique preparado para detetar ameaças aos novos componentes adicionados.

Em relação à recolha de dados, um problema será a gestão das contas de utilizador que devem ser seguidas para recolher os dados que permitem criar os classificadores. Neste momento, as contas escolhidas têm de ser procuradas manualmente, sendo necessário observar os *tweets* publicados e verificar a sua relevância. Uma alternativa interessante a este método seria a introdução de um mecanismo de procura automática de contas de utilizador que publicassem ameaças de segurança. Outra opção seria recolher publicações a partir de qualquer conta e não apenas de um conjunto específico que publique informação relacionada. Não sabemos à partida quem publica informação útil para os nossos objetivos, por isso seria desejável conseguir ter acesso a todas as mensagens publicadas de forma fácil e contínua de modo a dar uma maior resposta às ameaças que vão aparecendo.

Neste trabalho estudámos a prospeção de dados apenas no Twitter mas sabemos que existem outras redes sociais que podem servir de fontes de informação de qualidade que quando exploradas, poderiam aumentar o campo de ação do trabalho aqui realizado, ao enriquecer os conjuntos de dados extraídos.

Um dos grandes problemas na recolha de dados de fontes OSINT é a redundância da informação. Usando o Twitter como exemplo, quando uma ameaça é descoberta, é possível que vários utilizadores anunciem essa ameaça. Este tipo de conteúdo duplicado seria corretamente classificado pela nossa abordagem, pois preenche os requisitos definidos na fase de treino. Contudo, quando uma ameaça ou vulnerabilidade é demasiado popular por afetar sistemas muito usados por todo o mundo (por exemplo, um *browser*), o analista depara-se com muita informação que deixa de ser relevante, pois a partir do primeiro alerta, já foi notificado o incidente. Estudos no sentido de conseguir detetar

informação redundante seriam do maior interesse para reduzir ainda mais o conjunto de informação apresentada. Uma abordagem promissora seria aplicar algoritmos de *clustering* nos resultados positivos, a fim de agrupar *tweets* semelhantes. Também seria útil alguma técnica para verificar se a informação publicada nas redes sociais não tenha sido produzida unicamente para enganar os nossos classificadores (como visto por Sabottke et al. [22]).

Também devem ser estudados métodos de atribuir automaticamente uma classificação positiva, sem passar por algum classificador, às publicações que contenham um formato específico e bem conhecido. Algumas contas do Twitter publicam vulnerabilidades com os identificadores CVE (*Common Vulnerabilities and Exposures*). Se for possível confiar nessas contas e nas suas publicações, pode ser possível adicionar um mecanismo que filtra imediatamente as publicações que contém um código CVE, sem ser necessário algum decisor.

Bibliografia

[1] S Haykin, *Neural networks and learning machines*.: Upper Saddle River: Pearson Education, 2009.

[2] André Correia, Pedro Ferreira, e Alysson Bessani, "Descoberta de Ameaças de Segurança Através do Twitter," in *INForum*, 2016.

[3] Gašper Hribar, Iztok Podbregar, e Teodora Ivanuša, "OSINT: A "Grey Zone"?," *International Journal of Intelligence and CounterIntelligence*, vol. 27, pp. 529-549, 2014.

[4] Quirine Eijkman e Daan Weggemans, "Open Source Intelligence and Privacy Dilemmas: Is It Time to Reassess State Accountability?," *Sec. Hum. Rts.*, vol. 23, p. 285, 2012.

[5] (2016) Twitter.com. [Online]. https://twitter.com/

[6] Isabelle Guyon e André Elisseeff, *An introduction to feature extraction*.: Springer, 2006.

[7] Jure Leskovec, Anand Rajaraman, e Jeffrey David Ullman, *Mining of massive datasets*.: Cambridge University Press, 2014.

[8] (2016) Word2Vec. [Online]. https://code.google.com/archive/p/word2vec/

[9] Tomas Mikolov, Kai Chen, Greg Corrado, e Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, e Jeffrey Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111-3119.

[11] Corinna Cortes e Vladimir Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273-297, 1995.

[12] Christopher JC Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, pp. 121-167, 1998.

[13] (2016, Junho) scikit-learn. [Online]. http://scikit-learn.org/stable/

[14] (2016) Mahout.apache.org. [Online]. http://mahout.apache.org/

[15] Dean Jeffrey e Ghemawat Sanjay, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, 2008.

[16] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, e Robert Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 2010, pp. 1-10.

[17] (2016) Spark.apache.org. [Online]. http://spark.apache.org/

[18] Matei Zaharia et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012.

[19] (2016) Spark.apache.org. [Online]. http://spark.apache.org/mllib/

[20] Léon Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nimes*, vol. 91, 1991.

[21] Alan Ritter, Evan Wright, William Casey, e Tom Mitchell, "Weakly supervised extraction of computer security events from twitter," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 896-905.

[22] Carl Sabottke, Octavian Suciu, e Tudor Dumitraş, "Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 1041-1056.

[23] Bernardo de Simas Gaspar Rodrigues, "Open-source intelligence em sistemas SIEM," Dissertação de Mestrado em Segurança Informática, Faculdade de Ciências da Universidade de Lisboa, 2015.

[24] Kalyan Veeramachaneni e Ignacio Arnaldo, "AI2: Training a big data machine to defend.,".

[25] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, e Rebecca Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the workshop on languages in social media*, 2011, pp. 30-38.

[26] Luiz Fernando Sommaggio Coletta, Nádia Félix Felipe da Silva, Eduardo Raul Hruschka, e Estevam Rafael Hruschka, "Combining classification and clustering for tweet sentiment analysis," in *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, 2014, pp. 210-215.

[27] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, e Tiejun Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 151-160.

[28] Alec Go, Richa Bhayani, e Lei Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, p. 12, 2009.

[29] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, e Ming Zhang, "Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1031-1040.

[30] Jessica Elan Chung e Eni Mustafaraj, "Can collective sentiment expressed on twitter predict political elections?," in *AAAI*, 2011, pp. 1770-1771.

[31] Yelena Mejova, Padmini Srinivasan, e Bob Boynton, "GOP primary season on twitter: popular political sentiment in social media," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 517-526.

[32] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, e Isabell M Welpe, "Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment," *ICWSM*, vol. 10, pp. 178-185, 2010.

[33] Takeshi Sakaki, Makoto Okazaki, e Yutaka Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 851-860.

[34] Eiji Aramaki, Sachiko Maskawa, e Mizuki Morita, "Twitter catches the flu: detecting influenza epidemics using Twitter," in *Proceedings of the conference on empirical methods in natural language processing*, 2011, pp. 1568--1576.

[35] Pete Burnap, Walter Colombo, e Jonathan Scourfield, "Machine classification and analysis of suicide-related communication on twitter," in *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, 2015, pp. 75-84.

[36] Eva Zangerle e Günther Specht, "Sorry, I was hacked: A classification of compromised twitter accounts," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, 2014, pp. 587-593.

[37] Hyeoncheol Lee, Beomseok Hong, e Kwangmi Ko Kim, "Documents topic classification model in social networks using classifiers voting system," in *Proceedings of the 2015 Conference on research in adaptive and convergent systems*, 2015.

[38] Kathy Lee et al., "Twitter trending topic classification," in *2011 IEEE 11th International Conference on Data Mining Workshops*, 251-258, p. 2011.

[39] Inoshika Dilrukshi, Kasun De Zoysa, e Amitha Caldera, "Twitter news classification using SVM," in *Computer Science & Education (ICCSE), 2013 8th International Conference*, 2013, pp. 287-291.

[40] Delip Rao, David Yarowsky, Abhishek Shreevats, e Manaswi Gupta, "Classifying latent user attributes in twitter," in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, 2010, pp. 37-44.

[41] Freek Boutkan, Bachelor Opleiding Kunstmatige Intelligentie, e TM Kenter, "Building a rudeness classifier: a word-based approach," 2015.

[42] Xianghan Zheng, Zhipeng Zeng, Zheyi Chen, Yuanlong Yu, e Chunming Rong, "Detecting spammers on social networks," *Neurocomputing*, pp. 27-34, 2015.

[43] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, e Virgilio Almeida, "Detecting spammers on twitter," in *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, 2010, p. 12.

[44] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, e Suku Nair, "A Comparison of Machine Learning Techniques for Phishing Detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, 2007, pp. 60-69.

[45] (2016) Tweepy.org. [Online]. http://www.tweepy.org/

[46] (2016) Dev.twitter.com. [Online]. https://dev.twitter.com/rest/public

[47] (2016) Apache Spark - Cluster Mode Overview. [Online]. http://spark.apache.org/docs/latest/cluster-overview.html

[48] (2016) Introduction to Support Vector Machines - OpenCV 2.4.12.0 documentation. [Online]. http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[49] Twitter. "Number of monthly active Twitter users worldwide from 1st quarter 2010 to 1st quarter 2016 (in millions)". [Online]. https://www.statista.com/